Titre: Title:	Large-Scale Assortment Optimization
Auteur: Author:	Hugo Palmer
Date:	2016
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Palmer, H. (2016). Large-Scale Assortment Optimization [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2379/

UNIVERSITÉ DE MONTRÉAL

LARGE-SCALE ASSORTMENT OPTIMIZATION

HUGO PALMER
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
DÉCEMBRE 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

LARGE-SCALE ASSORTMENT OPTIMIZATION

présenté par : PALMER Hugo
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. ROUSSEAU Louis-Martin, Ph. D., président
M. LODI Andrea, Ph. D., membre et directeur de recherche
M. JENA Sanjay Dominik, Ph. D., membre et codirecteur de recherche
M. CHARLIN Laurent, Ph. D., membre

ACKNOWLEDGEMENTS

I would first like to thank my research director, Prof. Andrea Lodi. His continuous support during this year of research, as well as his confidence, have allowed me to push my research in the right direction. He allowed this thesis to be my own, but I must acknowledge that nothing would have been possible without his help and expertise. I am grateful to the program of the Canada Excellence Research Chairs (CERC), which supported our research, as well as the Institute for Data Valorization (IVADO). Being the witness of the development of Montreal as a world Data-Science flagship has been a unique opportunity, and has been a constant stimulation for me as a student.

I would also like to thank Prof. Sanjay Dominik Jena, my research co-advisor. His door was always open, and countless times I have saved days of research thanks to his help and availability. In particular, during the stressful moments of preparation of the conference and writing of the thesis, he has done all his possible to help me find the proper balance between comprehensiveness and conciseness. Being his first student after his appointment as a professor has been an honor.

This research would not have been possible without the support of the team of JDA Labs. In particular, I would like to express my gratitude to Marie-Claude Côté, manager of the team, as well as to Pawan Kumar Singh and Gabrielle Gauthier-Melançon. Marie-Claude has shown a significant interest in this research project, and I hope that some of my theoretical results will be of use one day in their industry. Pawan has provided me with high-quality data sets, which have saved me hours of data collection; and Gabrielle shared with me some of her business experience.

I would also like to thank those who shared my concerns during this period. In particular, to Adrien, Benoit, Cyril, Eva, Greta, Loïc, Lucie, Lucile, Perrine, and Thomas. Moreover, thanks to my office colleagues, Giulia Zarpellon notably, who have shared my work hours and the ups and downs of my research.

My parents, my sister, as well as my family in general have been continuously supportive during my studies, and encouraged me in my choice of crossing the Atlantic ocean, which they successively did to visit me here. Finally, I must express my gratitude to Roxane Morel, which not only has supported me during this year of research, but has also been proactively seeking for discussions about it. A substantial part of my theoretical results originate from our talks, and my research has largely benefited from her econometric knowledge. If she had been a professor, she would have deserved to be my third supervisor.

RÉSUMÉ

Nous présentons dans ce mémoire une solution au problème d'optimisation d'assortiment. Ce problème consiste à choisir, au sein d'un ensemble de produits, le sous-ensemble générant le revenu espéré le plus élevé pour le détaillant. Le fait qu'il existe un nombre exponentiel d'assortiments de produits possibles rend ce problème difficile-et intéressant. En effet, l'énumération explicite des assortiments est impossible en pratique dès que l'on dépasse la vingtaine de produits.

Notre but est de fournir un procédé d'aide à la décision pour l'optimisation d'assortiment basé sur les données. Dans cette perspective, la première étape naturelle est de proposer un modèle permettant de rendre compte des comportements des consommateurs. Nous nous intéressons aux modèles de choix basés sur les classements des produits, qui consistent à apprendre une distribution de probabilité sur l'ensemble des classements possibles des produits. En effet, la structure de ces modèles de choix permet une modélisation efficace du problème d'optimisation d'assortiment, dans la forme d'un problème d'optimisation linéaire. Les solveurs classiques permettent de les résoudre pour des instances de taille industrielle. Cependant, le principal inconvénient de ces modèles de choix est leur lenteur d'apprentissage. Ceci est dû au nombre factoriel de classements envisageables. L'apprentissage repose, dans l'état de l'art, sur un modèle de génération de colonnes dont la taille de l'espace des classements rend l'apprentissage du modèle de choix particulièrement long.

Pour accélérer l'apprentissage du modèle de choix, nous avons remarqué qu'il était possible, moyennant une adaptation du modèle de choix, de munir l'espace des classements d'une structure forte, permettant de hiérarchiser les classements dans un arbre de décision. Ainsi, nous avons défini un nouveau modèle de choix reposant sur la notion d'indifférence : plutôt que de ne considérer que des séquences classant chacun de produits, nous acceptons aussi les classements partiels de produits. Ceci signifie que nous nous autorisons à considérer des comportements de consommateurs qui, à-partir d'un certain point, sont indifférents entre les produits restants. Cette relaxation de la définition d'une séquence de produits nous permet d'accélérer considérablement la recherche de nouvelles séquences, dans l'algorithme de génération de colonnes, grâce à la structure d'arbre de l'espace. En effet, à chaque itération de la génération de colonnes, nous n'avons qu'à étendre l'arbre de décision d'un niveau supplémentaire.

Nous avons appliqué ce nouvel algorithme à des données réelles, fournies par notre partenaire industriel, ainsi qu'à des données artificielles. Dans les deux cas, nous observons des gains de temps de calcul d'au moins un ordre de grandeur. Ainsi, nous devenons capables de résoudre

des problèmes jusqu'à un millier de produits, alors que les algorithmes de l'état de l'art étaient limités à quelques dizaines.

Ensuite, nous avons étudié une extension possible de ce problème, qui consiste à inclure des nouveaux produits sur-lesquels nous n'avons aucune donnée de transaction, dans l'optimisation d'assortiment. Ces nouveaux produits ne sont connus que par leurs caractéristiques communes avec les anciens produits, ceux sur-lesquels nous avons des données de vente. Nous avons proposé une manière d'introduire ces nouveaux produits dans le modèle de choix évoqué plus haut. Cet algorithme repose sur une mesure de similarité entre les anciens et nouveaux produits et permet de généraliser des comportements de consommateurs connus sur les anciens produits à l'ensemble des anciens et nouveaux produits. Cette extension est particulièrement intéressante pour le problème du choix d'assortiment d'une saison de vente à l'autre, puisqu'il faut prendre des décisions sur les produits de la nouvelle collection à inclure sans avoir encore observé de ventes.

Par ailleurs, nous avons montré comment notre nouveau modèle de choix s'adapte au problème d'optimisation d'assortiment présent dans la littérature, ainsi qu'une manière de limiter le sur-apprentissage.

Nous présentons enfin des résultats expérimentaux montrant l'intérêt de l'optimisation d'assortiment. Notre programme d'optimisation, sur des cas réels, propose systématiquement des assortiments plus larges (c'est-à-dire avec plus de produits) que ceux observés dans la réalité. Nous montrons ainsi l'impact d'offrir un large choix au consommateur sur le revenu espéré par le détaillant.

Cependant, afin de prendre en compte les contraintes opérationnelles ainsi que la taille limitée des magasins, nous montrons que, même en incluant une contrainte limitant la capacité de l'assortiment à ce qui est observé dans les données, les revenus prédits restent plus élevés d'environ 35% à ce qui est observé dans l'ensemble de contrôle. Ceci montre donc l'impact de l'assortiment sur le choix du consommateur, mais aussi sur le revenu du détaillant.

Pour finir, nous proposons une extension de notre travail pour l'élaboration d'un logiciel d'aide à la décision pour l'optimisation d'assortiment. Les dirigeants de boutiques peuvent en effet être réticents à confier l'ensemble de la décision d'optimisation d'assortiment à un logiciel, et préférer garder cette responsabilité. Il peut donc être intéressant de leur proposer un outil suggérant des modifications à la marge d'un assortiment qu'ils proposent eux-mêmes, ainsi que de leur permettre d'évaluer l'impact prédit par notre approche de chaque modification de leur assortiment.

ABSTRACT

This thesis is concerned with assortment optimization. The problem of assortment optimization consists in choosing, among a set of n products available to the retailer, the subset that is most likely to result in the highest revenue. However, the exponential number of possible assortments makes it intractable to evaluate all of them, when twenty or more products are available.

In the perspective of data-driven assortment optimization, we propose to learn the transaction data, based on the ranking-based choice models. These models assume that customer preferences can be modelled as ranked sequences of the products. The intrinsic structure of such a choice model allows an efficient formulation of the assortment optimization problem, in form of a mixed-integer optimization problem that can be easily solved by mathematical solvers. Nevertheless, the training of such choice models scales poorly with the number of products. Classical approaches are based on a column generation algorithm to deal with the factorially large number of possible product sequences. In practice, this is intractable even for small numbers of products. We therefore propose a modification of the choice model and of its training.

This modification consists in allowing indifference between products after a certain rank: this is justified by the fact that after a certain rank, the ranks are meaningless. With this new formulation, we can structure the factorially large space of sequences in the shape of a tree, which makes it possible to decrease significantly the time required to learn the model. Indeed, at each iteration of the column generation, we only branch the tree to one more degree.

We have applied the new way of training the choice models to real store data provided by our industrial collaborator, and to artificially generated data. In both cases, we noticed that computation times decreased by at least an order of magnitude. We were also able to solve instances up to one thousand products, while previous approaches were limited to several dozens.

Besides, we have developed an extension of our algorithm, able to consider adding new products in the optimized assortment. We assume that we have no transaction data on those new products, but only common characteristics (called features) with the old products, for which we have access on transaction data. To deal with those new products, we introduce a measure of similarity among products. Then, we show a way of generalizing the behavior of consumers known among the old products to the set of old and new products. This extension is useful for the problem of season-to-season optimization of assortments, where products of the new season are to be inserted in the assortment without prior information about transactions among them.

Additionally, we have found a way of using our choice model in the optimization problem: we show how to limit the error of over-fitting with a boosting method.

Finally, we show experimental results indicating the interest of assortment optimization. On real instances, our process designs systematically assortments wider than what was noticed in real stores: this shows that, for the given industrial data, it is important to provide the consumer with a broad choice of products to ensure the retailer a higher revenue.

Nevertheless, being aware of the operational constraints that retailers face, such as the maximum quantity of products to be displayed at the same time, we have also run our experiments with a capacity constraint in the optimization problem. It consists in limiting the total number of products to be displayed in an assortment to what is observed in real stores. In this case, we still predict a 35% increase in revenues based on our choice model.

Last but not least, we propose an extension of our work for the design of a decision support tool for retailers. Indeed, we know that some store managers may be reluctant to delegate the whole process of assortment optimization to a fully integrated software. Hence, it may be worthwhile to leave the manager in the position of choosing himself which assortment of products he may like to expose while providing him with some insights on the predicted consequences of certain choices. In particular, what would be the impact on the average revenue per customer of adding this product to the assortment?

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

$[\![1,5]\!]$	The set on integers containing 1, 2, 3, 4, and 5.
CG	Column Generation
CG-GDT	Column Generation with the Growing Decision Tree
CG-LS	Column Generation with the Local-Search Heuristic
CM	Choice Model
GDT	Growing Decision Tree choice model
GEV	Generalized Extreme Value choice model
GT	Ground Truth choice model
IIA	Independence of Irrelevant Alternatives
IT	Information Technology
MAE	Mean Absolute Error
MIO	Mixed-Integer Optmization problem
MNL	Multinomial Logit choice model
MMNL	Multi-class MultiNomial Logit choice model
NL	Nested-Logit choice model
PLD	Product Line Design
PLSE	Product Line Software Engineering
RB	Ranking-based choice model
RC	Reduced Cost

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

The possibility to store and process large amounts of data has allowed new applications of Operations Research. Most companies have understood the potential of leveraging big data technologies to gain productivity.

The retail sector is a particularly competitive market in which very different players take place. Independently-run stores may compete with big retailers franchising hundreds of stores across the world. The former, typically led by an owner-manager, use its independence as an asset to prove customers the originality of their purchase, while the latter benefits from the lower prices offered by the buying group.

All of them have to make some decisions about the products they want to display to their customers. This decision is often made to ensure a sufficiently broad choice to clients while meeting the standards of the store: complying with the quality that the customer expects to find in this store. Nevertheless, big retailers may centralize those decisions. They have therefore to deal with:

- Putting a lot of different products into the shelves, therefore ensuring a broad panel of products to the consumer, which results in a high **conversion rate**: the potential customers entering the store are likely to find a product that they like
- Restricting the choice to some high revenue products, and hope that customers are willing to substitute from the low-revenue to the higher revenue products, which results in an increase in sales.

Most of the store managers solve this trade-off by experience: they have a good idea what the customers are willing to find, and aim at maximizing the revenues of their store. Those managers have a very accurate experience-based knowledge of the problem, but their experience is limited to the sales they have witnessed in their particular store. For example, some choices made by colleagues in other cities may have resulted in better results, and they might be interested in understanding which approaches were most successful.

Therefore, the availability of data in several hundreds of stores allows designing data-driven models that can catch the influence of assortment on the buying decision. In particular, we focus our work on understanding the **substitution effect**, which consists of the behavior of consumers facing the absence of their preferred product: some customers will buy another product, and others will leave the store without any purchase. To address this problem, we begin our work by finding a way of modeling the choices of consumers: this is the concept of **choice modeling**.

Specifically, the problem of assortment optimization is critical for season-to-season choice of products: retailers have to order their products some weeks before the beginning of the new sales season. For this application, they also have to consider new products on which they have no transaction data to exploit. Therefore, we want to propose a model able to be generalized to new products.

In this Master thesis, we explore some algorithmic-based insights that may help store managers in choosing the best possible assortment of products to exhibit to their customers.

We conducted this study in partnership with JDA Labs, the research laboratory of JDA Software, a company providing software for the retail industry. They were able to provide us with two anonymized data sets, including transaction and assortment data, which allowed us to apply our models to real industrial data.

1.1 Definitions and main concepts

The definitions below are valid throughout the entire thesis. We consider n different products that are available in the products catalog of the retailer: we represent the **set of products** as $[\![1,n]\!]$ (each integer between 1 and n corresponds to a product).

The problem of **assortment optimization**, which is the central objective of the thesis, consists of finding a subset of this set of products that will result in the maximum expected revenue. This problem, as we are going to see later, is NP-hard (there are 2^n different possible assortments for n products).

The *no-choice option* consists on the possibility that a given customer entering the store may end up not buying anything. We represent this possibility by the number 0, such as the set $[\![0,n]\!]$ represents the set of possible alternatives for a consumer entering a store. Indeed, he may buy a product (therefore the corresponding number in $[\![1,n]\!]$ is assigned to him) or not buy anything (therefore the number 0 is assigned to him).

For the sake of clarity, we are going to illustrate our theoretical notions with a small example of a very limited number of products: we chose six shoes representing six products in Figure 1.1

Figure 1.1 Small example of six products

An **assortment** is a subset of the set of products. We usually use the letter S to refer to an assort-

ment, possibly with a subscript when we consider several assortments. When we have to refer to a particular assortment, we use the subscript m for referring to a specific assortment, M being the number of assortments considered: $S_1, S_2, ..., S_m, ..., S_M$.

We show an example of two different assortments in Figure 1.2. The door, representing the *no-choice option* is always present in the assortments, meaning that the consumers may always decide to leave the store without any purchase.

Figure 1.2 Example of two different assortments

We define **old products** as products on which we have transaction, assortment and features data; while **new products** are products on which we only know the feature data. As we have briefly explained above, we want to generalize our choice models to new products on which we have no transaction data at all. To be able to do that, we will take into account the common characteristics between products: this way we may draw conclusions based on the fact that a particular new product looks like some well-known products, and therefore include this new product inside our choice model.

To formally represent the common points between products, we define a **feature** as a characteristic of the products. A feature can be a color, a texture, a pattern, a general level of quality, if the product is mainly for males or females, for which use it is designed (sportswear, dress, casual), the season during which the product is likely to be used, etc...

All those different types of features are categorical data that we may translate to binary variables (*Is the product brown?, Is the product red?, Is the product in leather?*, etc...). This allows us to define a **vector of binary features** characterizing a product. It will be easy to manipulate it because simple operations (L1-difference, scalar product) may help us to quantify formally the intuitive idea of how similar the products are.

We assume that each product returns a **revenue** that is constant over time. Depending on the priorities of the store manager, the revenue can be replaced by the **profit** (the revenue minus the cost of the product for the retailer) without loss of generality.

1.2 Problem definition

The field of retail planning has been boosted by the availability of **transaction data** at the retail group level which has been developed increasingly in the past years. With the development of Information Technology (IT) systems able to link all stores, warehouses, and engineering offices together in real-time, transaction data collection becomes standard, because it only necessitates storing a list of events in a file. This evolution has led to an intensive research on how to take transaction data into account to model the choice of customers to help retailers in their decision process.

In contrast, collecting **assortment data** is more complicated, since it consists in the evolution of the level of stocks of each product at each time. To collect assortment data, it is, therefore, necessary to insert a dedicated module in the Information Technology chain of the retail group. This explains why assortment data is still rare, and consequently why research on assortment optimization is still at its beginnings.

The main goal of our research is to **provide a framework of assortment optimization to retail groups**, adapted to their technical requirements, and based on past sales witnessed in their whole network of stores. The assortments considered **may take into account new products**, on which no transaction data is known but on which we have features, that is to say, common points with the well-known old products.

1.3 Research objectives

To solve this problem, we propose the following steps:

- Present a way of modeling the choice of consumers able to take advantage of the assortment data availability
- Consider new products and extend the model of choice to take them into account
- Find the optimal assortment based on old and new products
- Generate data to test the three first steps
- Use real industrial data to confirm the applicability of our approach to practical industrial requirements

1.4 Thesis outline

To answer those research objectives, we are going to begin with a literature review in Chapter 2, to present the state-of-the-art of our topics. Then, we will present the details of the solution in Chapter 3, which consists of three main points: choice modeling, then the introduction of new products, and assortment optimization in itself. Lastly, we will present the theoretical and experimental results in Chapter 4 that we get on both generated and industrial data to validate our approach.

CHAPTER 2 LITERATURE REVIEW

We now review the related literature of three sub-parts of the retail optimization process: choice modeling, assortment optimization, and generalization to new products. For a broader view of the assortment planning process, we refer the reader to the book chapter of Kök et al. (2008); and for an earlier view of the same problem, the literature review of Mahajan and van Ryzin (1999) can be useful.

2.1 Choice modeling

2.1.1 Parametric models

A parametric choice model is a model that can be described as a finite sequence of parameters $\{\theta_1, \theta_2, ...\theta_n\}$. McFadden (1973) has pioneered the research in the discrete choice model framework, and applied it a few years later to a practical case of residential location (McFadden, 1978). Since then, uncountable applications have been studied for those continuous models. The work of Ben-Akiva and Lerman (1985) applied the choice models to predict travel demand, and Guadagni and Little (1983) to scanner panel data. We refer to the work of Train (2009) for a complete overview of the topic.

The Multinomial-Logit (MNL)

The most classic parametric model is the Multinomial Logit choice model (MNL). Its primary interest is its high tractability both regarding estimation of parameters and evaluation of results.

However, it assumes certain undesirable hypothesis, like the Independence of Irrelevant Alternatives (IIA). This condition was first mentioned by Arrow in 1951 (Arrow, 1951) and detailed by (Ray, 1973). We can formulate it in a condensed form: "If an alternative x is chosen from a set T, and x is also an element of a subset S of T, then x must be chosen from S" (Sen, 1970). Several classic examples show that this axiom can be violated in very simple life choices (see for example the anecdote attributed to Sidney Morgenbesser (Hausman, 2011)).

The substitution effect is also very limited in the Multinomial Logit choice model since it is not possible to have two products with different substitution rates and equal penetration rate (see a formal counter-example in the work of Kök et al. (2008, Ch. 6, p. 110)). In practice, this limits the cases in which the use of the MNL provides sufficiently good results. In our studies, we will compare the accuracy of our models to the MNL, which will be seen as a baseline.

MNL-derived models

More sophisticated parametric models have therefore been proposed to take more complex human behaviors into account. The Multiclass Multinomial Logit (MMNL) is a multiclass extension of the MNL considering several coexisting consumer behaviors. We can also cite the Generalized Extreme value (GEV) of (McFadden, 1980) and its most used particular case: the Nested Logit (NL) (Williams, 1977). The Nested Logit gives a probability for outcome i that can be expressed as the product of two simple logits: the probability of being in the nest, and of being chosen assuming that i is in the nest. However, the concept of substitution is by nature limited to a small order. Several other extensions that can deal with some of the drawbacks of MNL or NL have been derived from McFadden's GEV model, such as the Paired Combinatorial Logit (PCL) (Chu, 1989), or even the Product Differentiation (PD) (Bresnahan et al., 1996). We refer the reader to Wen and Koppelman (2001) for more details on advantages and drawbacks of those various parametric choice models.

Locational model

The locational model proposed by Hotelling (1929) considers the consumers being present on a road segment represented by the segment $[0,1]$, and assumes that their utility function is a function of their distance to the stores: therefore, a customer chooses to go to the closest store from his house. Given a certain distribution of existing stores, Hotelling (1929) studies the choices a potential new competitor that has to decide whether or not to enter in the market.

This model considers all stores, products and prices being equal, the choice of consumer relying by hypothesis only on the *location of the store*. This limits its interest in practice. Nevertheless, this model in its simplest form can be transposed to study the impact of a change of one particular feature in a product on a continuous way. The main drawbacks are that all prices must be equal, and all other features should be the same, which limits its interest to markets where one particular feature has an important impact on the decision of the consumers (for example, the quantity of fat in a yogurt).

Exogenous demand models

In the exogenous demand models, each customer is supposed to have a favorite product. If this product is not present, then he will substitute to another product with probability d or will not buy with probability $1-d$ (the sale is therefore lost). The probability of substituting product i for product j is $a_{i,j}$. Smith and Agrawal (2000) use this model substitution in the sales and define an integer programming formulation for assortment planning, including the decision of optimal

stock planning.

This model is therefore suited to accommodate a simple substitution between products. Nevertheless, one could argue that some more sophisticated consumer behaviors could not be understood properly by this model, like higher-order substitution.

2.1.2 Non parametric choice models

As a substitute to parametric choice models, there also exist generic choice models that make fewer assumptions on human's behavior and can explain a wide variety of choice data. We will first introduce the notion of substitution that is critical for understanding some key aspects of human choices.

Consumer-driven substitution

In a retail context, the substitution effect can be described as the will of a consumer to buy a different product depending on the presence or absence of his supposed preferred product. There exist three main consumer-driven types of substitution (Kök et al., 2008):

- In daily shopping, a consumer does not find a product that he used to find; he decides to buy another one
- The consumer knows that his preferred product is supposed to be carried in the store (because he saw an advertisement), but it is not available the day where he visits and therefore buys another one
- The consumer does not previously know the article he wants to purchase, and therefore buys the article that gives him the highest utility among all items on the shelf.

The first case is called **stock out-based substitution**, and the two last consists in **assortment-based substitution**. In particular, the last case is typical of a one-time purchase, that we observe particularly for apparel.

Rusmevichientong's choice model: totally ordered data from consumers

Rusmevichientong et al. (2006) has proposed a general non-parametric choice model to solve an assortment planning problem. Nevertheless, it assumes to have access to some totally ordered customer preferences list, which is in practice not very common in the usual data sets (with the exclusion of the Auto Choice Advisor of General Electric, which motivated his article).

A ranking-based choice model

To consider more general transaction data sets, Farias et al. (2013) have proposed a generic choice model based on a probability distribution over the set of all possible rankings of the products. They demonstrate that their algorithm can converge to fit any ground truth transaction data, and optimize the algorithm to find the sparsest choice model fitting this data (namely, finding the simplest choice model that fits the data). They use the concept of parsimony, or the Occam's razor principle, which states that over several equally performant models, the simplest should be chosen.

A consumer behavior is defined as a permutation of the set of the products into the set of products. There are therefore $n!$ possible consumer behaviors, and the choice model is a probability distribution among those $n!$ permutations. Jagabathula (2011) shows that this choice model is general. Some results are summarized in Farias et al. (2013). Nevertheless, this model has a factorial number of parameters, and is, therefore, computationally intractable for industrial values of n (say, more than 10).

Evaluation of the ranking-based choice model

Bertsimas and Mišic (2016) recently proposed a framework based on (Farias et al., 2013) that includes a practical way of computing the choice model efficiently, with a column generation based approach. This approach is interesting in the way that it benefits from the generality of the non-parametric choice models and the evaluation of a full substitution effect. Besides, a heuristic is proposed to avoid solving the subproblem of the column generation to optimality, which allows evaluating in a reasonable time the choice model for up to 30 products for generated data. Nevertheless, they do not study the time needed to make the algorithm converge on real data, and for higher numbers of products. We will, therefore, propose in this paper a slightly different choice model, that allows being evaluated much faster, by understanding which part of the data can be explained by a distribution of the preferred products of each consumer, and which part of the data set needs substitution to be evaluated.

The choice model obtained by the column generation approach consists only of fully-ordered sequences of products. This approach is very general, but we have identified several drawbacks limiting its use:

- for real (hence noisy) data, the choice model evaluation is very long (see Jagabathula, 2011)
- for high numbers of products (say, more than 100), the evaluation is intractable: if a very

particular sequence is needed, the time to find it becomes huge.

- the choice model obtained is only one among many others that could have achieved the same objective value. In particular, after a certain rank, the rankings of the products is irrelevant. Therefore, there is no control on which rank is relevant in the choice model, which can pose problems for the decision maker.

The new choice model that we propose in this thesis allows keeping the generality of this choice model while circumventing these three drawbacks.

2.2 Assortment optimization

Assortment optimization consists of finding the subset S of the set of products N that provides the best expected revenue for the retailer. It necessitates a way of modeling the choice of the consumers, which has been discussed in the previous part, and the selection of a set of hypothesis.

2.2.1 Usual hypothesis

We list here the most common hypothesis: each can be assumed or rejected depending on the data type we are considering. We will in the following subsection consider different models that consider as valid a different subset of this hypothesis list:

- Operational costs are independent of the assortment
- The price of a given product is constant over the time (Rusmevichientong et al., 2014)
- All products have equal prices (Mahajan and van Ryzin, 1999)
- A product cannot be stocked-out due to high demand from consumers (Mahajan and Van Ryzin, 2001)
- All products belong to a single category and no inter-categories interaction are considered
- Consumers have not the possibility to compare with competitor stores (Cachon et al., 2005)
- There is a single selling period
- Assortment cannot be adjusted over time

It is inherently difficult to validate the expected revenue for a supposed optimal assortment because it is hard to translate recommendations into real business implementations, in a sufficiently high number of stores to check the expected revenue with a good confidence margin: there is little feedback on academic recommendations. Unfortunately, academic studies focus on analyzing past transaction data and assortments, but seldomly implement the theoretical recommendations of assortment into real store assortments.

The validation step of the algorithm is therefore limited to the validation of the choice model on the test set. Most of the articles cited in this section provide an estimation of revenues but have not implemented a validation for the assortment optimization phase.

The choice of the hypothesis is critical because it can dramatically change the complexity of an algorithm (which is important), but also the validity of the results (which is even more critical).

In this section we are going to review the most relevant articles proposing to design an optimal assortment, given the set of hypothesis we consider.

2.2.2 Assortment optimization with parametric choice models

Efficient assortment optimization with the MNL and MMNL

As we have explained in the previous section, the Multinomial Logit choice model is an interesting choice model because of its simplicity of prediction. It is, therefore, a fast model for evaluating the revenue predicted for a given assortment. Nevertheless, when considering the set of 2^n possible assortments, it becomes intractable to evaluate all the possible assortments.

Mahajan and van Ryzin (1999) have therefore proposed a way of computing the optimal assortment very efficiently, given a major hypothesis: that all products have the same price p. They rank the products in descending order of popularity; their utilities in the Multinomial Logit choice model model are therefore such as: $u_1 \geq u_2 \geq ... \geq u_n$. Then, they define the popular assortment set:

$$P = \{\varnothing, \{1\}, \{1,2\}, ..., \{1,2,...n\}\}$$

and show that the optimal assortment is in P. This result is interesting because it allows decreasing the number of evaluations from 2^n to $n+1$, making it highly tractable. Nevertheless, the hypothesis of unique price limits its use to very particular cases.

The hypothesis of unique price being very restrictive, some other works have been made to avoid it. Rusmevichientong et al. (2010) for example has proposed a framework for taking into account various prices while doing assortment optimization. Nevertheless, the MNL choice model may be unsuitable because of its drawbacks (see section 2.1.1): it is based on the mean

utility of the product to all customers, which is not enough complete when we want to consider customers with different profiles.

Bront et al. (2009) and Rusmevichientong et al. (2014) have shown that in the general case, the assortment optimization problem is NP-hard with the Multi-class MultiNomial Logit choice model. To solve this issue, Sen et al. (2015) proposes to use a conic integer programming approach to model the assortment optimization problem with the Multi-class MultiNomial Logit choice model, which is efficient when considering only capacity constraints, but which is limited to take other constraints into account.

For the Nested Logit choice model, Davis et al. (2014) has proposed some restrictive hypothesis that makes this problem solvable in a polynomial time.

A heuristic of local search to solve the problem in the general case with the MNL

Jagabathula (2014) proposes the algorithm ADXOpt (for Add, Delete, eXchange). Given a revenue function $R(S)$, which can be a black-box, the ADXOpt algorithm consists of adding, deleting, or exchanging products in the tested assortment to find a local optimum of the revenue. This algorithm is very fast and provides optimality gaps reasonable for unconstrained assortment optimization, but can show more disappointing results for constrained problem (for example a capacity constraint can reduce the optimality gap drastically, as demonstrated by Bertsimas and Mišic (2016)).

Assortment optimization with the Exogenous Demand choice model

The exogenous demand model considers that consumers arrive one after the other, and each makes his choice depending on the updated quantities in stock at the moment he arrives. For a given assortment, the aim is therefore to select the optimal quantities of each product to maximize profit. The probability that the m^{th} consumer chooses a given product is given by an expression whose exact computation is complex. Therefore, Smith and Agrawal (2000) define lower and upper bounds, which are tight and much easier to evaluate and use them to model the demand. The resulting optimization problem is a nonlinear integer program, which is therefore highly intractable for large n. To find some nearly optimal solutions, linearization is unavoidable.

Assortment optimization with the locational choice model

As explained in the section detailing the locational choice model, we consider here products with same prices and features, excepted a particular one that takes a continuous value in the

segment [0;1]. Gaur and Honhon (2006) have proposed an assortment optimization under the locational choice model. The authors separate the study into the two main types of substitution:

- Static substitution: a consumer makes a choice given the assortment (for example in reading a leaflet), but do not substitute if the product is stocked-out due to high demand from previous consumers (in which case the sale is lost)
- Dynamic substitution: a consumer makes a choice in the store, given the assortment of products not already stocked out.

Gaur and Honhon (2006) propose a systematic way of dealing with the case of static substitution and present a heuristic to take dynamic substitution into account. Nevertheless, the hypothesis of same prices among products is very strong and makes this approach not interesting for most cases.

2.2.3 Various effects on assortment optimization

Dynamic substitution

The problem of designing optimal inventory levels is known as the newsvendor model, referring to the dilemma of a newspaper seller that has to solve a trade-off between the number of copies to carry to maximize profits in limiting the potential losses. It has first been formulated by Edgeworth (1888).

Therefore, we consider the additional possibility of stock-out due to high demand in our assortment optimization problem. Hence, the problem becomes: find the optimal assortment and inventory level that maximizes revenues. We assume a certain function of cost that depends on the inventory levels. We are now interested in dynamic substitution, which happens when the preferred product is stocked-out, and the customer has to substitute for a less preferred option. Mahajan and Van Ryzin (2001) have studied this problem, and come to the conclusion that the retailer should stock more of the most popular products than what a traditional news vendor analysis would suggest.

Consumer search

Considering that a consumer will always take the best option that is presented to him can be somewhat restricting, in particular in highly competitive environments such as e-commerce or malls where several stores are present on the same market. One could consider the possibility of consumers to compare between stores. Cachon et al. (2005) extend the model of Mahajan and

van Ryzin (1999) by considering the possibility for consumers not to buy an acceptable option because they want to go and explore other stores. Cachon et al. (2005) show that it may be interesting to stock more preferred products, to prevent from this effect.

Online stores may often change their assortment, re-optimizing it each time where new data is available, depending on the reaction of new consumers. (Ferreira and Goh, 2016) has studied the interest for stores to change often their assortment and has shown that under some assumptions (mainly, the fact that the consumers are uncertain on the new products to be exhibited in the future assortments), this results in higher revenues. That is what the authors call the *value of concealment*.

2.2.4 Assortment optimization with non-parametric choice models

The ranking-based choice models present very interesting properties in term of assortment optimization. Indeed, they allow presenting linear mathematical programming formulations for most cases. In particular, Bertsimas and Mišic (2016) propose a Mixed-Integer Optmization problem (MIO) problem that find the optimal assortment corresponding to a ranking-based choice model. They define x_i as 1 if the product is included in the assortment and 0 else, and y_i^k is 1 if the option i is chosen under the consumer's behavior k in the assortment defined by the vector x. The revenue generated by option i is noted r_i. The optimization problem writes:

$$
\begin{aligned}
\max_{x,y} \quad & \sum_{k=1}^{K} \sum_{i=1}^{n} r_i \lambda^k y_i^k & \\
\text{s.c.} \quad & \sum_{i=0}^{n} y_i^k = 1 & \forall k \in \{1,...,K\} \\
& y_i^k \leq x_i, \forall k \in \{1,...,K\} & \forall i \in \{1,...,n\} \\
& \sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k \leq 1 - x_i, & \forall k \in \{1,...,K\} \forall i \in \{1,...,n\} \\
& \sum_{j:\sigma^k(j)>\sigma^k(0)} y_j^k = 0 & \forall k \in \{1,...,K\} \\
& x_i \in \{0,1\}, & \forall i \in \{1,...,n\} \\
& y_i^k \geq 0, \forall k \in \{1,...,K\}, & \forall i \in \{0,1,...,n\}
\end{aligned}
$$

They also present some constraints that can be added in order to take some operational requirements into account, such as:

- Lower bound L on the size of the assortment: $L \leq \sum_{i=1}^{n} x_i$
- Upper bound U on the size of the assortment: $\leq \sum_{i=1}^{n} x_i \leq U$

- Lower (L_S) and upper (U_S) bounds on the size of some subset S of products: $L_S \leq \sum_{i \in S} x_i \leq U_S$
- Precedence constraints: if i if included, then j must also be included: $x_i \leq x_j$

The structure of the problem imposes the variables y to be binary. Therefore there is no need of adding a constraint stating that they should be binary. Hence, this MIO problem has only n non-continuous variables, making it tractable even for large n. Computational results of Bertsimas and Mišic (2016) show that for values of n up to 30, the computation time for the constrained MIO problem is below 1 second.

2.3 Product line design and generalization to new products

2.3.1 Product line design

The assortment optimization, as we have seen in the previous part, consists in selecting among a set of products the subset allowing the best expected revenue. On the other hand, the set of products among which we have to choose is a concept only valid for retailers that do not participate in the design of the products themselves. Based on consumer preferences, a retailer may want to design himself the products before considering to put them on the shelves. Optimizing the assortment given a continuous set of products (that we consider being on a *line*) is called Product Line Design.

Mussa and Rosen (1978) have studied this concept by assuming that products can be described by their intrinsic quality and their prices. From those two features, they show what is the optimal solution for a retailer willing to maximize its profits. They assume convex production costs in the quality. Moorthy (1984) explains for this problem of Product Line Design the differences between a monopolist and a competitive behavior.

Tang et al. (2004) study the interest of encouraging the consumers to book before the selling season, for an example of perishable products (fruits). They show that the retailer can benefit from an "Advance booking discount" program and present a way of computing the optimal discount price.

The concept of Product Line Software Engineering, explained in (Lee et al., 2002), is an industrial application of Product Line Design. It is based on the idea that it is more efficient to design an entire collection rather than all the products one per one.

2.3.2 Parametric choice models

Parametric choice models such as MNL-derived models can be easily adapted to make predictions on features and not on products. To do that, the parameters learned by the choice model can be based on features of the products, instead of the products themselves. This idea comes from Lancaster (1966), who stated that the utility is derived from the features of the products, and not from the products themselves.

The most classic way of dealing with that is to define a set of features $(f_1, f_2, ..., f_F)$ of the products. For $p \in \{1, ...f\}$, the corresponding feature f_p must take exactly one label: $l_p^1, l_p^2, ..., l_p^{n_p}$. For example, when considering cars, the features can be the color, type (SUV, urban car, ...), number of passengers (5 or 7), number of doors, brand and motor power.

Notice that we only consider discrete features, contrary to the locational choice model that only assumes one continuous feature. It is also possible to convert the encoding of features into a binary vector: $\{b_p^k\}$, with $p \in \{1, ..., F\}$, and $k \in \{1, ..., n_p\}$. b_p^k is equal to *True* when f_p equals l_p^k, and *False* else.

Instead of considering the products themselves during the training phase of the algorithm, it is possible to consider the features. This way, a product is represented by its vector of features. It is also possible to consider non-linearities of the features by combining them two by two: for example, an SUV with very low motor power is unlikely to be chosen. It is, therefore, possible to keep track of virtually all non-linear effects between features, but leads to computationally inefficient computations: MNL-derived models are computed by gradient descent, which requires a high number of iterations to converge. When we take all those effects into account, the complexity surges (up to w^2 for all the pairs of features, or w^3 for all triplets of features, etc... where w is the size of the binary vector b).

The flexibility of use of the parametric choice models is, therefore, counter-balanced by the higher computation times required to train the models.

2.3.3 Non-parametric choice models

On the contrary, non-parametric choice models use products as indivisible entities. For the example of ranking-based choice models, it is impossible to change the view, from a product-based model to a features-based model, as we have exposed in the previous subsection. Jagabathula has developed some interesting frameworks for non-parametric choice models and exposes some ideas for generalizing ranking-based choice models to new products in the conclusion of his Ph.D. thesis (Jagabathula, 2011). One is to find the features that best explain the rank

lists exhibited in the choice model. Another is to us the choice model as a prior in the Bayesian sense, and infer new behaviors based on Bayes' rule.

Nevertheless, this assumes that all the ranks are meaningful, and our experiments show that after a certain rank, the rankings are meaningless. We will interest ourselves on this problem in the following parts: this is why we decided to focus our work on the ranking-based choice models. Moreover, as we have seen, the paper of (Bertsimas and Mišic, 2016) has paved a way in this direction, in providing us with the mixed-integer assortment optimization problem. We are therefore going to go further with the formality of the ranking-based choice models.

CHAPTER 3 METHODOLOGY OF THE SOLUTION

As we have seen, the framework developed in the papers of Jagabathula (2011), Farias et al. (2013) and (Bertsimas and Mišic, 2016) has several advantages. We will refer to the model of Bertsimas and Mišic (2016) as **CG-LS**, because it is based on Column generation with a Local Search heuristic. We are going to present in this chapter the details of the approach that we have found useful to approach the problem.

We will first introduce the new choice model that we have designed and discuss its advantages and drawbacks. Then we will present our method for taking into account new products. Lastly, we are going to show how our approach can address the problem of assortment optimization, which was the main intention of our work.

The process, from data to optimization, is summarized in Figure 3.1: we consider as data a training and a test sets, and the features of both old and new products. The data is used at all three steps of the process: training choice models, then integration of new products, and assortment optimization. Those three steps will be the next three sections of this chapter.

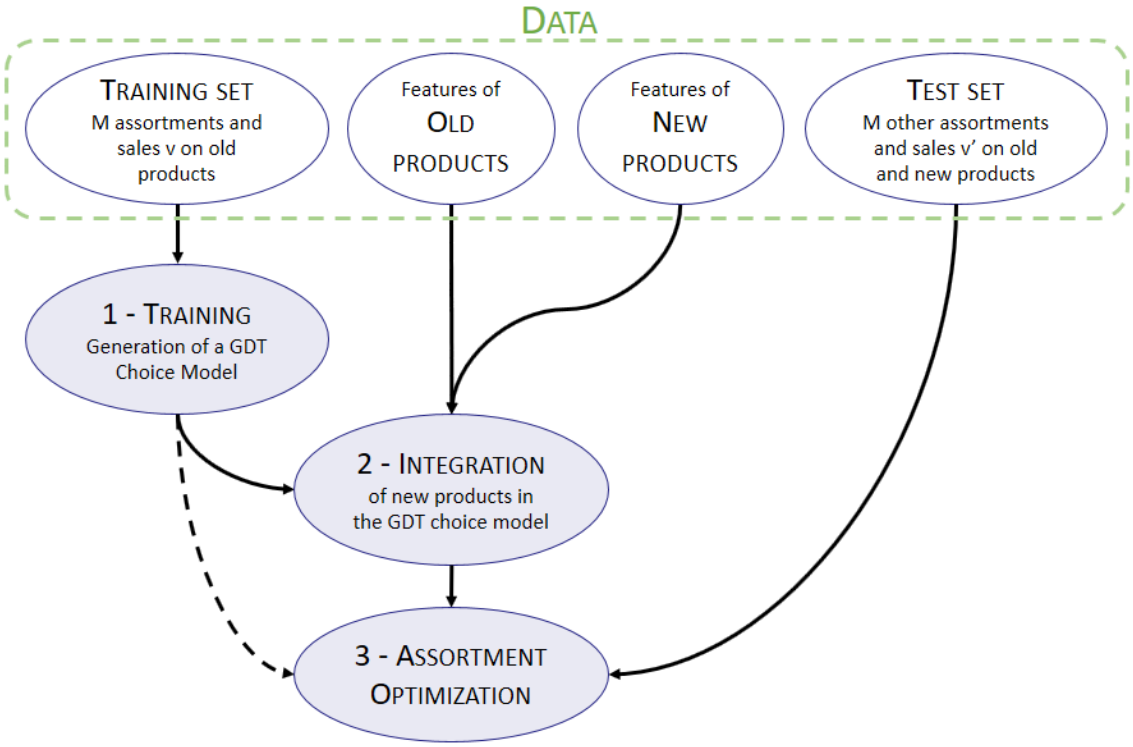

Figure 3.1 Summary of our data-driven approach in three steps

3.1 Training choice models

Choice modeling can be described as a way of providing models coherent with choice data. In our case, the choice data consists of transactions made by consumers in retail stores, and depend on several parameters such as availability, price, and features. Choice models usually deal with those three parameters, but most of them do not take substitution between products into account. We will, therefore, present choice models that are specially designed to handle substitution: the **ranking-based choice models**.

The data that we used in this work consists of transaction data and assortment data. The latter is the list of all products present in a given store at a certain time that we may aggregate at a week or day level. The main interest of the ranking-based choice models is to take advantage of the assortment data when it is present, which is not possible with usual models. Hence, we are going to see to what extent an assortment can influence the revenue.

3.1.1 How to take into account the assortments and the sales?

The data that we consider in this thesis consists of lists of transactions with time-stamps as well as the set of products exhibited at a given moment. Therefore, we can convert the list of transactions into a vector of probability of sales for each assortment, showing the probability distribution that a random customer entering the store ends up by choosing each product. That is what we call the **vector of actual sales** v. In Figure 3.2, we show two examples of assortments, and the sales distribution on it. We note the presence of the no-choice option, represented by a door, in all assortments.

Figure 3.2 Actual relative sales v on two assortments

The training set is composed of M assortments and the sales on each of them among the n products: the vector of size $M \times n$, whose component associated with the probability of selling the product i to a random customer when the assortment S_m is exhibited is: $v_{i,m}$. The test set consists of sales on M other assortments.

3.1.2 Learning the choice model: state of the art

Consumer's behavior

We define a consumer's behavior as a strict order of preferences among the options. Figure 3.3 shows an example of a consumer's behavior. We chose to represent the no-choice option by an open door, representing the possibility for a consumer to leave the store without purchase.

We assume that a consumer described by this sequence has a deterministic behavior: she will buy her preferred product among those present in the assortment. In the presence of her preferred product (the brown sport shoes) she will buy it; when it is not present, she will buy her second preferred product (the black sport shoes) if it is present; and when both her two preferred products are not present, she chooses the third option, which is the no-choice option: the sale is lost. We may notice that because the no-choice option is always present, the rankings after it are meaningless.

Figure 3.3 An example of consumer's behavior: ranking among all the products

Consumer's behaviors are a complete way of modeling the choice of consumers, because it can predict the way a particular consumer is going to behave when any assortment is exhibited to her.

More formally, we use the same definition as Bertsimas and Mišic (2016): a customer's behavior is a bijection $\sigma^k : [\![0,n]\!] \longrightarrow [\![0,n]\!]$ such that each option is given a rank: for $i \in [\![0,n]\!], \sigma^k(i)$ represents the rank of preference of product i. The preferred product is therefore the product i such as $\sigma^k(i) = 0$.

Defining a choice model

As exemplified above, a consumer's behavior can be seen as a permutation of the set of options $[\![0,n]\!]$. There exist therefore $(n+1)!$ different consumer's behaviors for n products. We represent the set of all consumer's behaviors in a tree as exemplified in Figure 3.4 for $n = 3$. Each branch represents a consumer's behavior; the preferred product is the closest to the root.

Jagabathula (2011) defines a choice model based on the theory of consumer's behaviors. It consists in a probability distribution (λ_k), $k \in [\![1,K]\!]$, among the set of all consumer's behaviors.

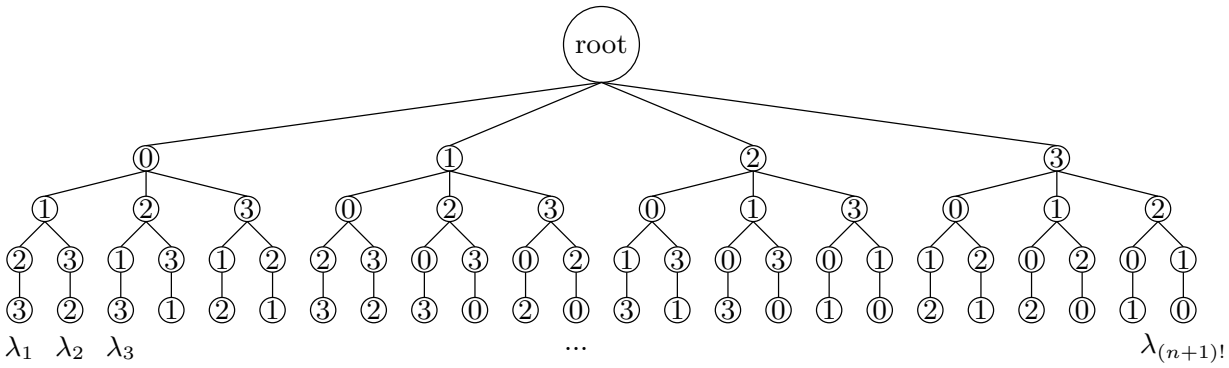

Figure 3.4 Probability distribution among the set of all possible consumer's behaviors

We can understand the tree as: the weight λ_k associated to a particular consumer's behavior represents the proportion of potential consumers whose behavior is described by σ_k.

For instance, a proportion λ_1 of all potential customers have the branch (preference list) $(0,1,2,3)$. The no-choice option 0 being present in all assortments, they will always be able to pick their first choice each time: 0. Moreover, the consumers represented by the branch $(1,3,0,2)$ will take the product 1 if it is present; else, the product 3 if it is present; and if both 1 and 3 are not in the assortment, then they will choose the no-choice option 0.

Using the choice model for prediction

For a given set of assortments $\{S_m\}$ and a set of consumer's behaviors $\{\sigma_k\}$, we define the matrix of choices A as in the paper of Bertsimas and Mišic (2016):

$$A_{i,m}^k = \begin{cases} 1 \text{ if } i = \operatorname{argmin}_{j \in S_m \cup \{0\}} \sigma^k(j) \\ 0 \text{ otherwise} \end{cases}$$

More intuitively, a component of the matrix $A_{i,m}^k$ is equal to 1 if and only if the product i is the highest ranked product among the products present in the assortment S_m according to the consumer's behavior σ_k:

$$A_{i,m}^k = \begin{cases} 1 \text{ if } i \text{ is chosen by consumer } k \text{ among assortment } S_m \\ 0 \text{ if } i \text{ is not chosen by consumer } k \text{ among assortment } S_m \end{cases} \tag{3.1}$$

Based on this definition, we consider the vector $A\lambda$, whose components are given by:

$$\forall i [\![0,n]\!], \forall m \in [\![1,M]\!], (A\lambda)_{i,m} = \sum_k A^k_{i,m} \lambda^k$$

The scalar $(A\lambda)_{i,m}$ represents the probability of selling the product i to a random customer entering the store when the assortment S_m is exposed. It is exactly this prediction of sales $A\lambda$, that we want to be as close as possible to the vector of actual sales $v_{i,m}$. Therefore, we aim at minimizing the L1-error defined as:

$$|A\lambda - v| = \sum_i \sum_m |(A\lambda)_{i,m} - v_{i,m}|$$

Evaluation of the choice model

To evaluate this choice model, we have to find the vector λ that realizes the lowest L1-error. Bertsimas and Mišic (2016) propose a linear formulation to find the vector λ, which we will call the *Master problem*:

$$\begin{aligned} \min_{\lambda,\epsilon^+,\epsilon^-} \quad & 1^T\epsilon^+ + 1^T\epsilon^- & \\ \text{s.t.} \quad & A\lambda + 1^T\epsilon^+ - 1^T\epsilon^- = v & (\alpha) \\ & 1^T\lambda = 1 & (\nu) \\ & \lambda, \epsilon^+, \epsilon^- \geq 0 & \end{aligned} \tag{3.2}$$

This program returns the probability distribution λ (its components sum to 1 and are non-negative) that achieves the lowest objective value, therefore minimizing the L1-error defined above.

As we have seen, a *column m* of the matrix of choices A^k represents the decisions of a particular consumer's behavior on the set of assortments $\{S_m\}$. We may, therefore, consider only a subset of the possible consumer's behaviors, and solve the master problem. This means that we may not achieve the lowest L1-error because some major consumer's behaviors may be missing in the subset considered, but with the Master problem defined above we are still able to catch a sufficiently good value for this subset of columns.

Column generation: identifying relevant preference sequences

Once the master problem solved with a certain subset of the columns in the matrix of choices A, we may be interested in finding new columns to concatenate to A in order to lower the L1-error.

To do this, we use the technique of Column Generation CG-LS. When the master problem has been solved, we can use the dual variables α and ν associated to the constraints at optimality, and define the reduced cost rc for a potential new column a to add:

$$rc(a) = -\alpha a - \nu$$

In the theory of mathematical programming, the reduced cost of a column can be described as the potential decrease in the objective value if we can increase the new component λ_{k+1} associated to this new column by one unit, while the other components of λ do not change. Nevertheless, because of the constraint of the components of λ summing to 1, increasing the value of the component λ_{k+1} results in decreasing by the same amount other components of the vector λ, which may lead to a smaller decrease in the objective value. Based on those considerations, at each iteration of the column generation procedure, the objective value may decrease or stay constant, and the best decrease expectancy is reached for the lowest reduced-costs columns.

We can summarize the column generation procedure as the following:

1. Choose randomly a consumer's behavior
2. Solve the master problem and get the dual variables α and ν
3. Find a new column achieving a negative reduced cost
4. Add this new column to A and go to 2, unless the stop criterion is achieved.

The step 3 is in practice the most time-consuming. Indeed, as we have seen before, we have to select a particular column in a factorially-big space ($(n+1)!$ columns in total for n products). Two approaches are considered. The first one consists in solving a subproblem that finds the lowest reduced-cost columns to add, based on constraints ensuring that the columns respect the structure of the problem. This approach is mathematically exact, but very complex in practice, as pointed out by Bertsimas and Mišic (2016). Therefore, the authors propose an alternative approach that allows to find quickly negative reduced costs columns with a local search heuristic.

Optimality is reached when all potential new columns have a positive reduced cost: it means that no column may decrease the objective value again. Nevertheless, we may accept a less restrictive stop criterion, such as to be close enough to optimality.

In Figures 3.5, 3.6, 3.7 we have plotted some iterations of the algorithm of column generation for our small set of 3 products. At iteration 8 (Figure 3.5), we have eight columns in the matrix

of choices A. Then, we find the column with a negative reduced cost (Figure 3.6) and add it to the matrix of choices. Finally, (Figure 3.7), we obtain a new probability distribution among nine columns instead of eight.

Limitations of the approach

The local search heuristic proposed by Bertsimas and Mišic (2016) allows finding optimal solutions for higher number of products than when solving the subproblem to optimality. In particular, they propose computations for up to 30 products. Nevertheless, our implementation of the heuristic of local search, as it is described, shows that computing times increase very quickly when we increase the number of products up to 60 or even higher orders of magnitude.

The approach that we have described consists mainly in two steps: solve the master problem, then find a new column with the heuristic of local search. Our experiments show that the latter is the limiting step. Indeed, the space in which it has to find a column is of cardinality $(n+1)!$, which increases factorially. Hence, we proposed an alternative way of stating the problem of finding a new column that shows much better training times and allows to go to larger numbers of products.

3.1.3 Taking advantage of the tree structure: the Growing Decision Tree choice model

So far, we have described the consumer's behaviors as particular elements uncorrelated one to the other. Nevertheless, it is easy to find a structure among them, in the shape of a tree. Beginning at the root, there are $n+1$ branches corresponding to the $n+1$ options available as possible preferred products; then, for each particular branch, we can repeat the procedure with the n other possibilities. By continuing $n+1$ times, we can build all the possible permutations featuring the consumer's behaviors.

Our idea is to exploit this tree structure to learn the model more efficiently. We are going in this part to introduce a slightly different choice model, based on the model CG-LS, that can be learned much more quickly.

Sensibility to the first products

In the model CG-LS, a consumer's behavior corresponds to a fully ordered sequence of the products. Nevertheless, in our computational experiments, we noticed that in practice, only the five or ten first products are meaningful.

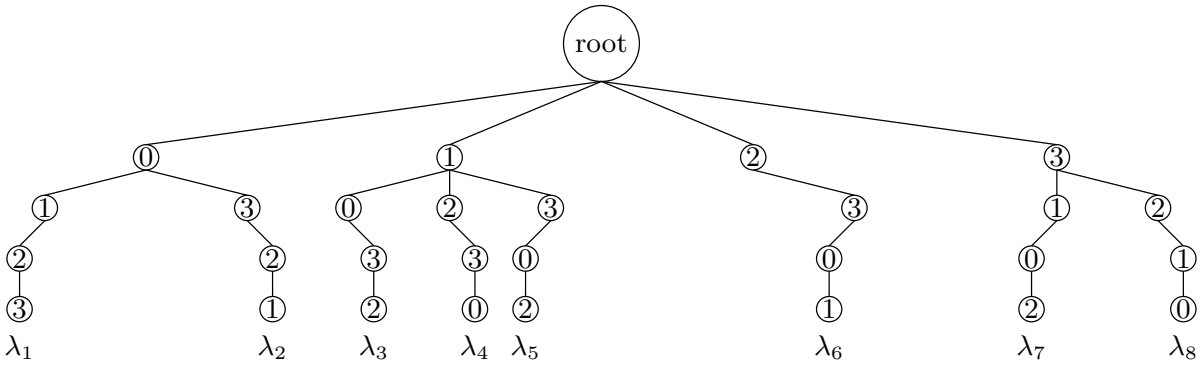

Figure 3.5 Iteration 8: finding the optimal probability distribution $(\lambda_1, ..., \lambda_8)$

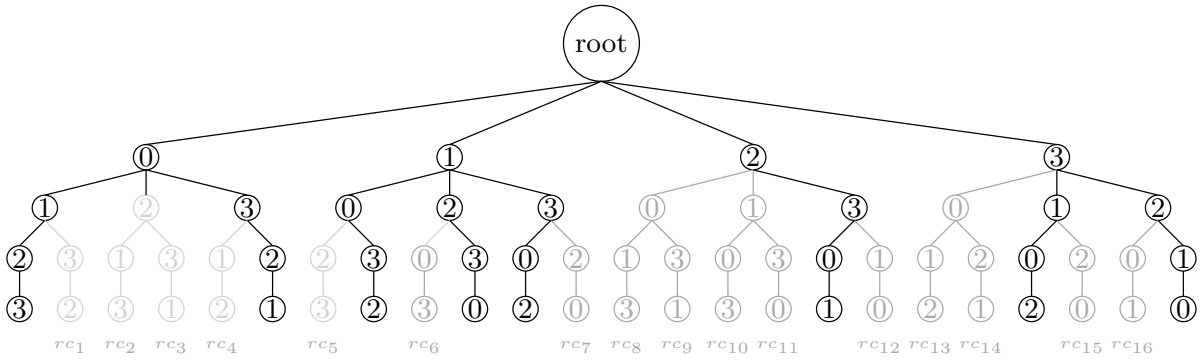

Figure 3.6 Iteration 8: computation of the reduced costs and selection of the lowest

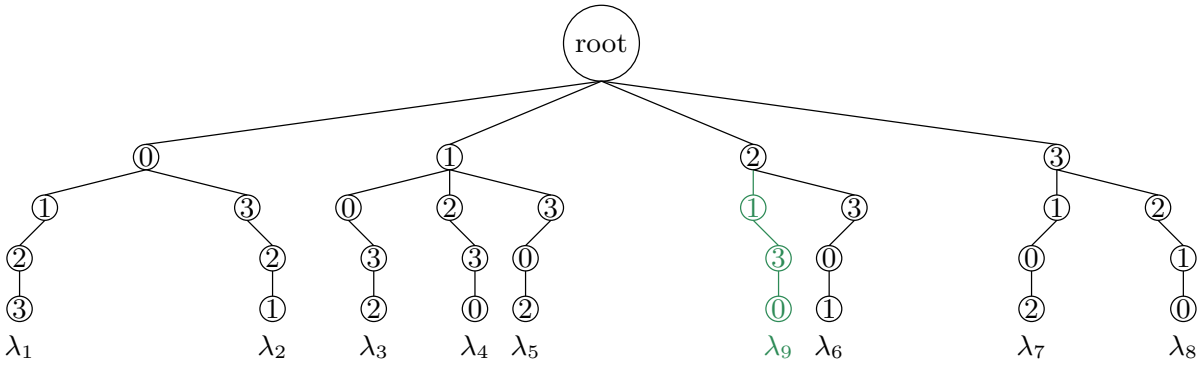

Figure 3.7 Iteration 9: finding the optimal probability distribution $(\lambda_1, ..., \lambda_8, \lambda_9)$

Example Let's call s the average sparsity of the assortments, and assume that each product is equiprobable. Then the probability that the product i in position $\sigma(i)$ be chosen is: $s^{\sigma(i)} * (1-s)$ (the probability of not having every single option with a better rank than i, and having i in the assortment). For instance, for a sparsity of $s = 0.5$, the product ranked at the tenth position will be chosen with a probability of $\frac{1}{2^{11}} = 0.05\%$, which is negligible.

Therefore, for non-sparse assortments, the probability that a low-ranked option is selected is exponentially low: the chosen choice model gives a tremendous importance to the first products of each permutation.

The natural conclusion of this is that the choice model CG-LS as it is designed conveys much more information than needed: in an example of 100 products, and a choice model of 200 permutations, we could only remember the 10 first products of each permutation without loss of information in most cases. Moreover, the 90 low-ranked products of each permutation may not explain the sales in the training set, but be falsely used to predict the sales. This could cause bad prediction accuracy because of overfitting.

For those reasons, we decided to change the definition of the consumer's behaviors slightly.

A new definition for consumer's behavior: allowing indifference between products

Now, we allow some indifference in the consumer's behaviors: after a certain rank (which may be between 0 and n), we consider that we have no more information on the rankings of the other products.

In Figure 3.8, we show on the left the previous definition of a consumer's behavior in the model CG-LS, where all products are ranked, and on the right the new definition, where here only the two first products are ranked, and the five other options are not: the consumer is supposed to be indifferent to them. We can represent it with the symbol ||, meaning that the order after the || has no meaning.

For a consumer's behavior σ, we define the **set of preferred products** as the set of ranked products, on which we have ranking information. We note $n_{pref}(\sigma)$ the number of ranked products. Similarly, we define the **set of indifferent products** as the set of products to which the customer is indifferent (it is composed of $n - n_{pref}(\sigma)$ products)

Practically, instead of representing a consumer's behavior by a permutation of $[\![0, n]\!]$, consumer's behaviors will now consist in rankings of products from 0 to $n_{pref}(\sigma) - 1$ and $n - n_{pref}(\sigma)$ prod-

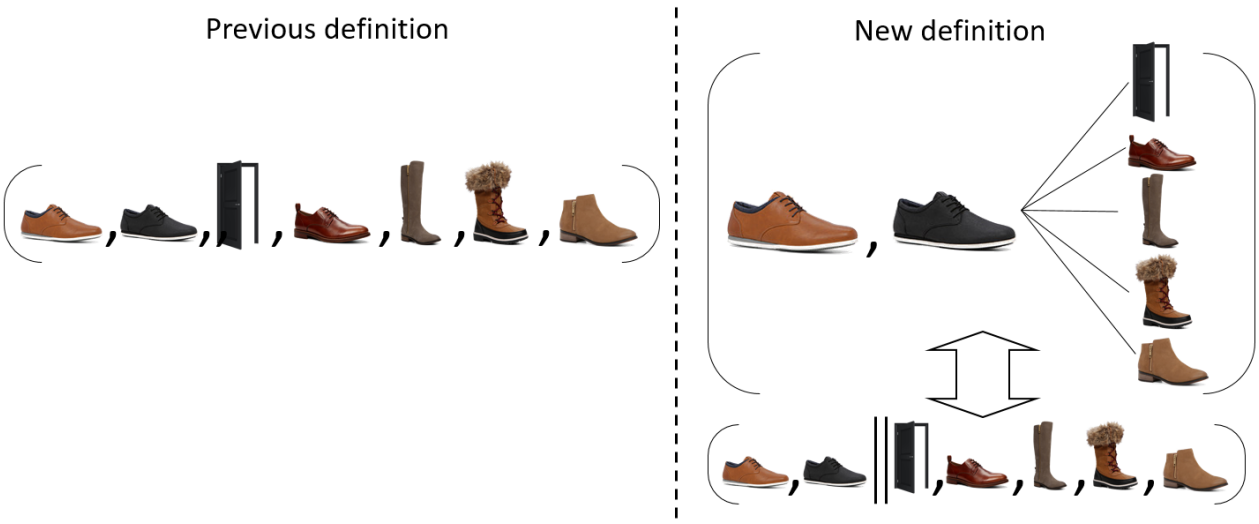

Figure 3.8 A new definition of consumer's behavior to allow indifference between products

ucts whose value is set to $n-1$:

$$\sigma(i) = \begin{cases} \text{rank of i} & \text{if i is in the preferred products} \\ n-1 & \text{if i is in the indifferent products} \end{cases}$$

To summarize the modification, we can say that a consumer's behavior is represented by:

- An entire branch in CG-LS
- A leaf in the Growing Decision Tree choice model

Indeed, a node in the tree means that we have the information about the preferred products (the nodes between the root and where we stop); the indifferent products are the others, by deduction.

Matrix of choices

With this new definition of a consumer's behavior, we are still able to define a matrix of choices: for a given set of assortments $\{S_m\}$ and a set of consumer's behaviors $\{\sigma_k\}$, we extend the matrix of choices A as the following:

$$A^k_{i,m} = \begin{cases} 0 & \text{if } (i \notin S_m) \text{ or } (i \in S_m \; and \; \exists j \in S_m - \{i\}, \sigma^k(i) > \sigma^k(j)) \\ 1 & \text{if } i \in S_m, \forall j \in S_m - \{i\}, \sigma^k(i) < \sigma^k(j) \\ \dfrac{1}{|S_m|} & \text{if } i \in S_m \text{ and } \forall j \in S_m, \sigma^k(j) = n-1 \end{cases} \tag{3.3}$$

More intuitively, the third option means that when no preferred product is present in the assortment, then instead of having a 1 for the chosen product, the sales are going to be split into all the products present in the assortment, each of them being assigned a probability $\frac{1}{|S_m|}$, where $|S_m|$ is the number of products in the assortment.

With this definition, the relation of structure of the matrix of choices still holds:

$$\forall (k,m), \sum_i A^k_{i,m} = 1$$

It means that for each consumer's behavior and among each assortment, the sum of probabilities of choosing the options is equal to 1.

Adaptation of the column generation procedure

With this new definition of the matrix of choice, we are still able to compute the vector of predicted sales $A\lambda$ as before:

$$(A\lambda)_{i,m} = \sum_k A^k_{i,m} \lambda^k$$

Therefore, we can keep the same master problem with this definition of the matrix of choices. The column generation procedure still consists of the same four steps, but the third step (find a column achieving a reduced cost) is changed to benefit from the modification of the choice model.

Definition - σ^2 is a **sub-behavior** of σ^1 if all the products ordered in σ^1 are ordered with the same ranks in σ^2.

Definition - The **sub-behaviors of rank 1** of a behavior σ^1 are the sub-behaviors that have exactly one more ranked product. For example, the sub-behaviors of rank 1 of $\sigma = (4,1,4,0,4)$ are : $(2,1,4,0,4)$, $(4,1,2,0,4)$, and $(4,1,4,0,2)$. Similarly, we define the sub-behaviors of rank h of σ as the sub-behaviors that have h more ranked products.

With those definitions, we can split a behavior into several behaviors that are more accurate,

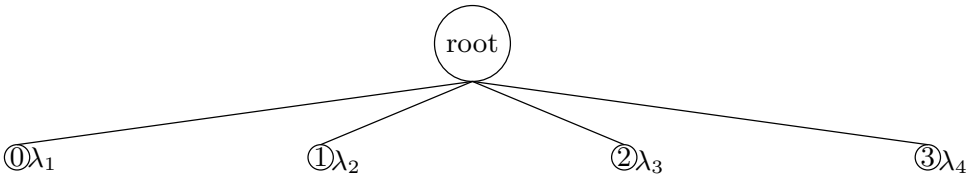

Figure 3.9 Iteration 1: finding the optimal probability distribution $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$

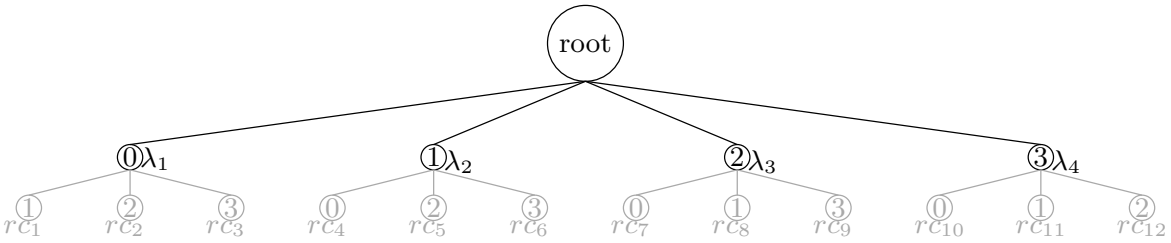

Figure 3.10 Iteration 1: computation of the reduced costs and selection of the lowest

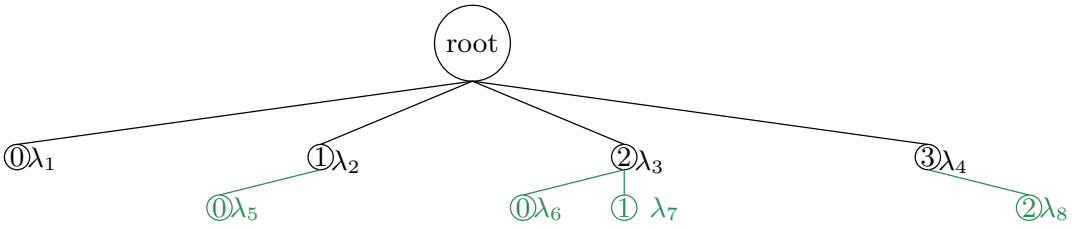

Figure 3.11 Iteration 2: finding the optimal probability distribution $(\lambda_1, ..., \lambda_8)$

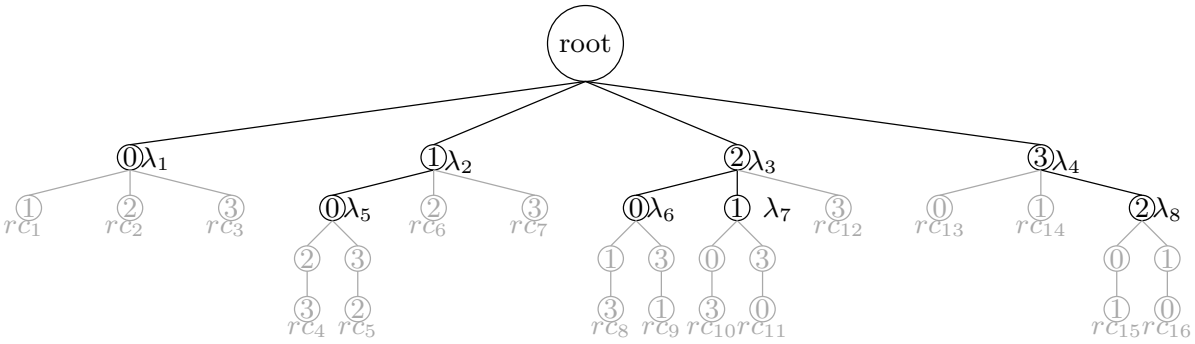

Figure 3.12 Iteration 2: computation of the reduced costs and selection of the lowest

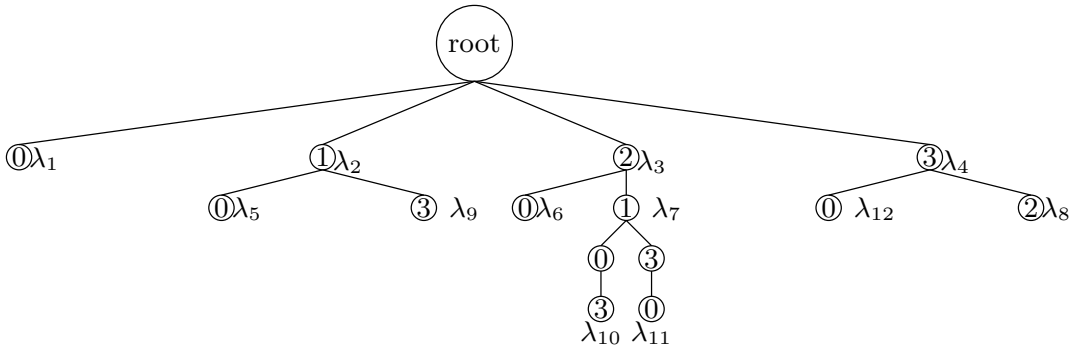

Figure 3.13 Iteration 3: finding the optimal probability distribution ($\lambda_1, ..., \lambda_12$)

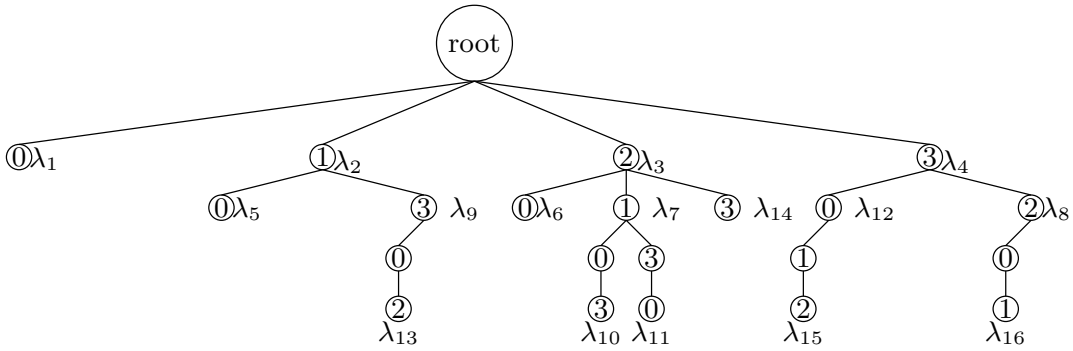

Figure 3.14 Iteration 4: Training finished

allowing to have a higher level of substitution between the products. Indeed, when none of the preferred products is present in the assortment, instead of considering that the probability $1/|S_m|$ is affected to all the products, we can split this probability more accurately to fit the training set better.

To find columns with low reduced cost to add to the matrix of choice, we operate as follows: we list all the sub-behaviors of rank 1 of all the columns of the matrix of choice A; we compute their reduced costs and select the smallest of them. It is possible to select several of them at the same time, which in practice made the algorithm a little faster. We denote s the number of columns to add at each iteration (in the process pictured above, we had $s = 4$ because we added 4 new consumer's behaviors at each iteration.

We showed in the Figures 3.9, 3.10, 3.11, 3.12, 3.13, 3.14 an example of training of the GDT choice model. In Figure 3.9, we have the first iteration where we insert one consumer's behavior per product, without substitution for the moment. Then (in Figure 3.10), we compute the reduced costs of all the sub-behaviors of rank 1, which is in practice computable because there are less than n^2. We repeat this two times, and we finish in a few iterations with a fully trained choice model.

Computational complexity - Let $C(A)$ be the number of columns of the matrix A at a given iteration, and n the number of products that we consider. Each of the $C(A)$ columns has a number of sub-behaviors of rank 1 that is equal to $n - n_{pref}(\sigma)$ (the number of products minus the number of preferred products in σ), which is in the worst case equal to n. Hence, this algorithm has $C(A) * n$ reduced costs to compute at maximum. In practice, we have most of the times $C(A) < 10 * n$, therefore, an overall complexity of $O(n^2)$ is needed (taking as reference the computing time of reduced costs, which is a scalar product hence linear in the number of products).

As a comparison, the CG-LS model does not provide an upper bound, because we may never find in the space of size $(n+1)!$ the proper column to add (achieving a negative reduced cost). In practice, their algorithm finds interesting columns at the beginning of the training, but when we necessitate very specific consumer's behaviors to improve the choice model, it may take excessively long computing times.

Therefore, at the end of the training phase of the GDT choice model, we have a choice model that consists of a list of consumer's behaviors (potentially partially-ranked) and a probability distribution among them. An example of choice model is given in Figure 3.15

Figure 3.15 Example of GDT choice model

Parallelization of the computation

Our adapted column generation procedure, therefore, consists of two main parts:

- Solve the linear master problem
- Select the s sub-behaviors of rank 1 to add to the matrix of choices A before the next iteration

Solving the master problem is achieved by mathematical solvers. In practice, they use all threads available, but the efficiency of the parallelization is not obvious and may depend on several factors. Thus, this first step may be poorly parallelizable. However, the step of computation of reduced costs is much more parallelizable because the columns of the matrix whose sub-behaviors of rank 1 are to be computed can be sent to different workers.

3.2 Integration of new products

In the previous section, we have shown how our Growing Decision Tree choice model can learn transaction data efficiently, from a set of assortments. Nevertheless, this choice model, as all ranking-based choice models (Farias et al., 2013), is based on products with historical sales data and therefore cannot intrinsically learn how to manage new products, on which we have no transaction data.

In this section, we are going to show how our choice model can be extended to deal with new products. Note that this is a context of very limited data, in the sense that we have to provide predictions based only on similarities between old products and new products. To evaluate this, we will define a metric between products.

3.2.1 Defining the features

Given a data set of products, we assume that all the categorical information of each product can be described as a vector of binary features f. For example, if the color is described by a categorical data of 5 different colors (blue, red, green, black, white), then the information can be easily transposed as a vector of five binaries: red would be $(0,1,0,0,0)$, and white $(0,0,0,0,1)$. If there is another categorical field (say, the texture for example) which allows three values: (leather, jeans, other), then we can consider two ways of modeling it:

- A vector of three binaries: $(1,0,0)$ would represent *leather*, while $(0,0,1)$ would represent *other*.
- A vector of two binaries, because we can consider that the *other* is not a feature in itself: $(1,0)$ would represent *leather*, $(0,1)$ would represent *jeans*, and $(0,0)$ would represent *other*.

Moreover, in the case of categorical data with only two possible outcomes, only one binary feature can be sufficiently: 0 and 1 would represent each outcome.

We are also able to model continuous variables, as a single value which can be useful for example to model the price bands. To do this, we have to normalize the data of this feature of the training set to ensure the same mean and variance for all continuous variables. Indeed, we want that all features have the same weight, which requires the normalization. For a given observation of variable x_i for each product i, the **standard score** is defined as:

$$z_i = \frac{x_i - \overline{x}}{sd(x)}$$

where $\overline{x}$ is the mean of the observations x_i and $sd(x)$ is the standard deviation of the distribution $\{x_i\}_i$. The standard score z conveys the same information than the original observation x, but has a mean of 0 and a standard deviation of 1. In our case, for consistency with the binary variables, we want a mean of $\frac{1}{2}$ and a standard deviation of $\frac{1}{2}$ (which is what is observed for binary variables where half of them take the value 1).

To transform directly a continuous observation series into another with a standard deviation of $sd_{targ} = \frac{1}{2}$ and a mean of $avg_{targ} = \frac{1}{2}$, we can use the formula:

$$z_i = \frac{sd_{targ}}{sd(x)}(x_i - \overline{x}) + avg_{targ}$$

We show an example of normalized series in Table 3.1. We may have values less than 0 of greater

than 1, but that is preferable than truncating them to be in a $[0,1]$ interval and loosing information. Moreover, it would be impossible to set up the mean, the standard deviation, the minimum and the maximum with a linear transformation, because this would set up 4 parameters while there are only two degrees of freedom in a linear transformation.

Table 3.1 Normalization of continuous variables

	x	**z**
obs1	5	0,12
obs2	10	0,19
obs3	20	0,32
obs4	35	0,51
obs5	100	1,35
mean	34,00	0,50
std	38,63	0,50

Finally, we can define the vector of features concatenating the vectors representing the categorical data and the continuous data. We, therefore, assume now and for the rest of the thesis that we have access to a vector of features for each product i: f_i.

3.2.2 Separability of the problem

The GDT choice model consists of a list of consumer's behaviors, and a probability distribution among them to give their relative importance. We are, therefore, able to study the problem separately for each consumer's behavior σ_k. Indeed, according to the choice model, a proportion λ_k of the consumers behave as described by σ_k. This **separability** is a significant advantage, because it is much simpler to study several small problems instead of a single big one.

For the rest of the section, we consider a single consumer's behavior σ^k associated to a probability of occurrence λ_k. The question for this part is, therefore: *How the customers represented by (σ^k, λ_k) on the set of old products are going to behave when also facing new products?*

3.2.3 Coherence between consumer's behaviors

Let P^0 be the set of old products, and P^N the set of new products. A customer behavior σ_{all} among all products is defined as **coherent** with σ^k if the consumer behavior σ_{all} preserves the order defined by σ^k on the set of old products:

$$\forall (i,j) \in P^0, \sigma^k(i) < \sigma^k(j) \implies \sigma_{all}(i) < \sigma_{all}(j)$$

We now consider a set of n old products and n_{new} new products. We note n_r the number of preferred products in σ^k, which is the number of products ranked at the beginning of the sequence.

We define $C(\sigma^k)$ as **the set of coherent customer's behaviors**. We denote $D = |C(\sigma^k)|$ the cardinality of this set, which is finite because there are less consumer's behaviors coherent with σ^k than the total number of permutations of $[\![0, n+n_{new}]\!]$ (which is $n+n_{new}+1$). The exact value of D can be expressed in function of (n, n_r, n_{new}), as shown in Annex A.

We can state that the customers described by (σ^k, λ_k) among the old products are going to be represented by a customer's behavior among all products that is coherent with σ^k: indeed, their preferences among new products are not known which means that their preferences among the new products have not been observed.

Therefore, we can state without loss of generality that the number of customers represented by λ_k are going to be split among all customers' behaviors coherent with σ_k that we note σ_k^c , with a certain new probability distribution $\{\lambda_k^c\}_c$ where:

$$\sum_{c \in [\![1,D]\!]} \lambda_k^c = \lambda_k$$

We, therefore, have to predict the value of λ_k^c to predict the way that consumers are going to behave in the presence of new and old products.

3.2.4 Definition of a metric among products to measure the proximity of products

To exploit the features data, we need a measure of similarity between the products. For two products i and j, we are going to count the number of categorical features that they have in common, and the absolute difference between the linear features. We also introduce a parameter γ that is a vector of the same size as the features vectors and composed by positive numbers that depict the relative weight of the features.

For a given vector of parameters γ, we can define the **distance between two products** as the following scalar product:

$$d(i,j) = \langle \gamma, |f_i - f_j| \rangle$$

d is a distance because the following properties are true (remember that the components of γ are positive):

- $\forall f_i, f_j, d(i,j) \geq 0$
- $\forall f_i, f_j, d(i,j) = 0 \Leftrightarrow f_i = f_j$

- $\forall f_i, f_j, d(i,j) = d(j,i)$ (symmetric)
- $\forall f_i, f_j, f_k, d(i,j) + d(j,k) \geq d(i,k)$ (triangle inequality)

Evaluation of the parameters $\gamma_1, \gamma_2, ...\gamma_F$ of the distance

As we have seen in Section 3.2.4, the distance is dependent on the value of the parameters γ as:

$$d(i,j) = \langle \gamma, |f_i - f_j| \rangle$$

We have explained at the beginning of this thesis (see Subsection 3.2.1) that we use normalized continuous features, to ensure the same weight in the distance in the case where all components of γ are set to 1. Nevertheless, we know that all features are not as important for consumers, and therefore it may be worthwhile to tune the parameters.

One way to tune the distance parameters is to train a classic MNL based on the features instead of the products. We are going to show this here.

Instead of learning the utilities of the products, we want to estimate the utilities of the features. We note the utilities of the features $u^{feat} = (u_1^{feat}, u_2^{feat}, ..., u_F^{feat})$. We assume that, for a product i of vector of features f_i, the utility $u(i)$ of the product i is given by:

$$u(i) = \langle u^{feat}, f_i \rangle$$

We, therefore, assume in this part a linear dependence on the utilities of the features to the utilities of the products. Then, we can define a MNL based on the features:

$$\mathbb{P}(i|S) = \frac{e^{u(i)}}{e^{u(0)} + \sum_{j \in S} e^{u(j)}}$$

This model will tune the parameters of the features u^{feat} to achieve the lowest possible training error. We may split our set into a training set and a test set to ensure the ability of the model to generalize. For this purpose, we may use the technique of cross-validation with 50% of assortments in the training set and the other half in the test set.

The output of the model is, therefore, a vector giving the utilities of the features. We notice in computational results that some features are linked to a null utility, which means that the model did not find an impact of this feature on the output.

3.2.5 Likelihood of a coherent permutation

We have defined in the previous parts the distance between two products and how to practically compute it. We have also explained the concept of coherence between consumer's behaviors: to generalize the choice model to new products, we have to learn the relative weights λ_k^c of each coherent consumer's behavior among all products σ_k^c with σ_k, with the constraint $\sum_{c\in[\![1,D]\!]} \lambda_k^c = \lambda_k$.

Our idea is to assign to each σ_k^c a *score* that shows how likely it is that a customer behaving like σ_k when only old products are exhibited to him behaves like σ_k^c among all the products. This score must be positive, and may take arbitrary high values; then, we will be able to convert the scores to the $\{\lambda_k^c\}$, for $c \in [\![1, D]\!]$.

Below, we are going to propose a way of computing the scores.

Computing the likelihood scores

For a certain k, we consider the set $C(\sigma_k) = \{\sigma_k^c\}_c$ of coherent customer's behaviors with σ_k. For a given $c \in [\![1, D]\!]$, some new products are ranked in σ_k^c: $R(\sigma_k^c)$, and some are not ranked: $[\![1, n_{new}]\!] \setminus R(\sigma_k^c)$.

For each new product i in $R(\sigma_k^c)$, we consider the product immediately before and the product immediately after. At least one of those two products exists:

- if there is no product ranked immediately before, it is because $\sigma_k^c(i) = 0$, i.e. i is ranked in first position
- if there is no product ranked immediately after, it is because $\sigma_k^c(i) = n_{pref}(\sigma_k^c)$, i.e. i is ranked in last position.

We define:

- i^+ as the product ranked immediately before i if it exists, or the product ranked immediately after if it does not exist.
- i^- as the product ranked immediately after i if it exists, or the product ranked immediately before if it does not exist.

With those definitions, $i+$ may be equal to i^-: when i is in first position (i.e. $\sigma_k(i) = 0$) or in last preferred position. We may then define the likelihood of i at its position in σ_k^c:

$$L(i,\sigma_k^c) = \frac{1}{d(i,i^+)} + \frac{1}{d(i,i^-)}$$

The closer (in terms of the metric defined above) the new product i is to its immediate previous and following products, the higher the likelihood function will therefore be. We can then sum over all the products in $R(\sigma_k^c)$ to get the total likelihood score of σ_k^c:

$$Sc(\sigma_k^c) = \frac{1}{|R(\sigma_k^c)|} \sum_{i \in R(\sigma_k^c)} L(i,\sigma_k^c)$$

This formula gives us the intuitive view that it is likely that customers behaving as σ_k among the old products are going to behave like the coherent consumer's behaviors in which the products inserted in the sequence look like the old products present in σ_k.

We represent this graphically in Figure 3.16: for $k = 1$, we consider the consumer's behavior σ_1 associated to the probability $\lambda_1 = 0.5$. We wrote as an example 3 coherent customer's behaviors, and how to compute the likelihood associated to the new product inserted. Because there are only one new product in the preferred products of the customer's behaviors we took as examples, the score is directly equal to the likelihood.

Figure 3.16 Example of computation of the likelihood score for 3 coherent consumer's behaviors

Conversion of the likelihood scores to $\{\lambda_k^c\}$

We want to transform the likelihood scores S_c into a distribution $\{\lambda_k^c\}$, under the constraint that they sum to λ_k:

$$\sum_{c \in [\![1,D]\!]} \lambda_k^c = \lambda_k$$

An intuitive way is to impose a linear transformation, which is the simplest possible. The solution is therefore:

$$\lambda_k^c = \lambda_k \frac{Sc(\sigma_k^c)}{\sum_p Sc(\sigma_k^p)}$$

Nevertheless, we could also use other regularization functions, such as the well-known softmax function:

$$\lambda_k^c = \lambda_k \frac{e^{Sc(\sigma_k^c)}}{\sum_p e^{Sc(\sigma_k^p)}}$$

The softmax function increases the relative weights of the highest scores, because of the exponential in its definition.

We have therefore a theoretical way of finding, from a choice model $\{(\sigma_k, \lambda_k)\}$ among the old products, a choice model among both old and new products: $\{(\sigma_k^c, \lambda_k^c)\}$.

Limiting the size of the new choice model In practice, the method described above returns a high number of columns in the choice model among all products. As we are going to see in the next section, the assortment optimization model computing time is very sensitive to the number of columns; hence, it may be interesting to limit them. To do this, we can erase the columns resulting to a too small value of λ_k ($\lambda_k \leq \tau_{min}$, with for example $\tau_{min} = 10^{-4}$), and normalize the other components of λ to still have a probability distribution.

3.3 Assortment optimization

In the previous parts, we have shown how to learn a GDT choice model, then how to integrate new products inside. In this section, we are going to demonstrate that this framework is particularly adapted to the problem of assortment optimization.

3.3.1 Assortment optimization in the model of Bertsimas and Mišic (2016)

We suppose that each option $i \in S$, the set of all products, is associated to a revenue r_i. We assume that we have computed with the previous section a choice model, composed of a set of

permutations (σ_k) and a distribution of probabilities among those permutations (λ^k).

As exposed in the literature review (see Section 2.2.4), Bertsimas and Mišic (2016) propose a computationally efficient way of finding the optimal assortment in the shape of a Mixed Integer optimization problem. We define the variable x_i as 1 if the product is included in the assortment, and y_i^k is 1 if the option i is chosen under the permutation k in the assortment defined by the vector x. We can, therefore, write the optimization problem as:

$$
\begin{aligned}
\max_{x,y} \quad & \sum_{k=1}^{K} \sum_{i=1}^{n} r_i \lambda^k y_i^k & \\
\text{u.c.} \quad & \sum_{i=0}^{n} y_i^k = 1 & \forall k \in \{1,...,K\} \\
& y_i^k \leq x_i, \forall k \in \{1,...,K\} & \forall i \in \{1,...,n\} \\
& \sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k \leq 1 - x_i, & \forall k \in \{1,...,K\} \forall i \in \{1,...,n\} \\
& \sum_{j:\sigma^k(j)>\sigma^k(0)} y_j^k = 0 & \forall k \in \{1,...,K\} \\
& x_i \in \{0,1\}, & \forall i \in \{1,...,n\} \\
& y_i^k \geq 0, \forall k \in \{1,...,K\}, & \forall i \in \{0,1,...,n\}
\end{aligned}
$$

This problem is stated for a choice model as described by the formalism of CG-LS, in which the consumer's behaviors are fully ordered. In the case of the GDT choice model, we have access to consumer's behaviors that are partially ordered, which means that we cannot run in a plug-and-play way the GDT choice model into the MIO.

Thus, we are going to present a way of adapting the GDT choice model to the formalism of CG-LS.

3.3.2 Adaptation of the GDT to the assortment optimization model

When the choice model has been trained to optimality

If the GDT choice model has been trained to optimality, then the matrix of choices A contains only binary variables. This means that all the training set is explained by the ranked products of each consumer's behavior. Therefore, we can choose randomly a sequence among the indifferent products of each consumer's behaviors, and transform the GDT choice model into a totally-ranked choice model. Then, we have an equivalent formulation and we may use the MIO exhibited above.

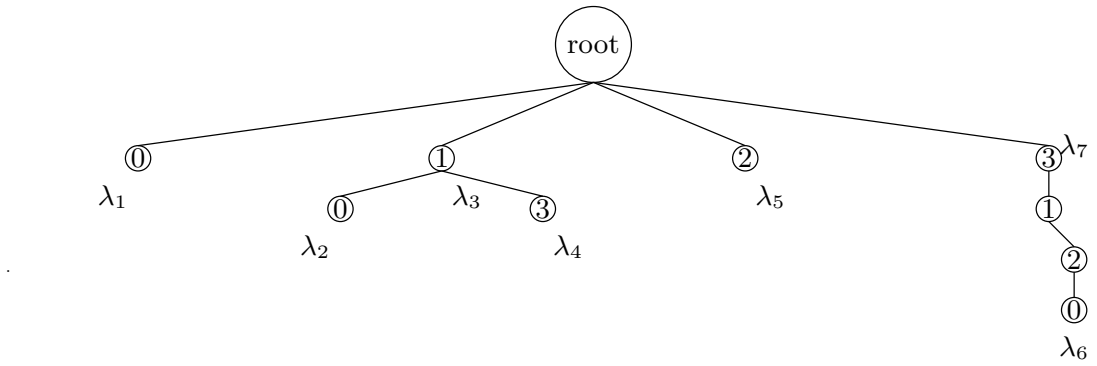

Figure 3.17 Example of GDT choice model

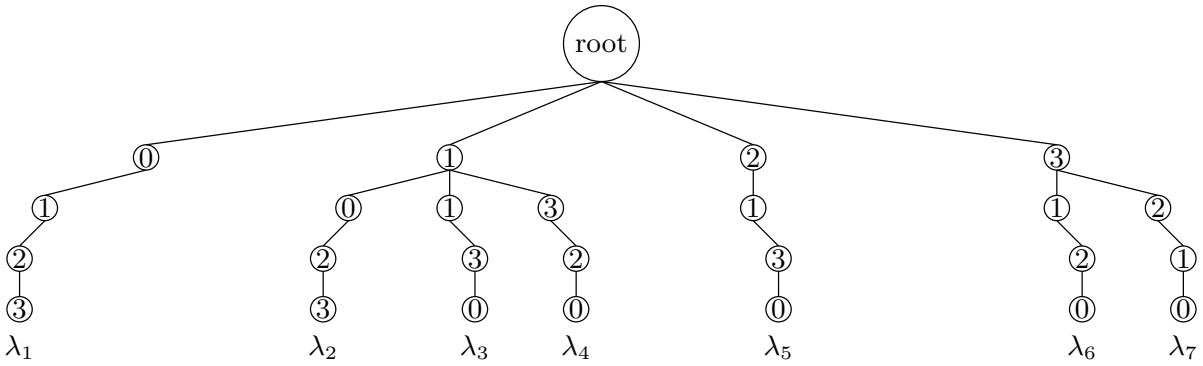

Figure 3.18 The GDT choice model converted in a fully-ordered choice model

When choice model has not been trained to optimality

Nevertheless, for industrial-size instances, we may not find the optimality in reasonable time, and it may be interesting to stop the computation without having branched the tree enough. In this case, we are going to show that there exists a theoretically equivalent formulation of totally-ranked choice model. Then, we will show how to practically accelerate the computation with a heuristic.

Equivalent formulation We consider a consumer's behavior σ_k from the GDT choice model, where $n_{pref}(\sigma_k)$ products are ranked at the beginning. There are $n+1-n_{pref}(\sigma_k)$ unranked products (which are the indifferent products). In the decision tree of the GDT choice model, we may consider all the consumer's behaviors whose representing node is located under the node representing σ_k: they are the **sub-behaviors** of σ_k. We denote this set as: $SB(\sigma_k)$

Theorem The choice model is invariant when replacing the tuple (σ_k, λ_k) by:

$$\left\{\left(\sigma_k^q, \frac{\lambda_k}{|SB(\sigma_k)|}\right), \forall q \in SB(\sigma_k)\right\}$$

Proof We are going to show that for a given assortment S, the products are given the same probabilities with both formulations. We consider an assortment S.

- If there exists a preferred product by σ_k in S, then the same product is chosen by σ_k and by all the consumer's behaviors in $SB(\sigma_k)$: then, this product is given the probability λ_k with the first choice model, and $\frac{\lambda_k}{|SB(\sigma_k)|} * |SB(\sigma_k)| = \lambda_k$. The other products are with probability 0 in both cases. (for example, in Figure 3.19, the product 2).
- If there does not exist a preferred product in S, then: with the first choice model, all products in S are chosen with probability $\frac{1}{|S|}$. With the second choice model, we notice that the problem is symmetric in all the products in S. Thus, we have the same probability of selling any product in S: then, all products in S are chosen with probability $\frac{1}{|S|}$.

To summarize, in all cases, we have the same probability of selling each product with both choice models: they are equivalent.

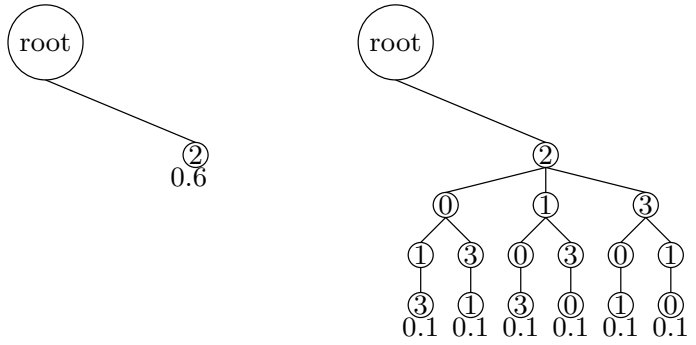

Figure 3.19 Equivalence of the two formulations

In a GDT choice model, we can, therefore, replace each consumer's behavior by its sub-behaviors as explained above: we have therefore a theoretically consistent way of transforming the GDT choice model in a choice model with fully-ranked consumer's behaviors. This allows us to plug the choice model in the Mixed-Integer Optmization problem of Bertsimas and Mišic (2016), and get the optimal assortment.

Acceleration of the computation: heuristic Nevertheless, this way leads to a final choice model with $(n+1)!$ consumer's behaviors, which makes it intractable for high valued of n. We can, therefore, propose a heuristic, able to limit the choice model to a reasonable number of columns while ensuring a sufficiently good prediction accuracy.

For a given σ_k, instead of inserting all its sub-behaviors, we may only select a random subset of them. The more we take, the higher the accuracy but the longer the computing times.

We, therefore, define a minimum number of sub-columns to insert for each original consumer's behavior: n_{min}. Moreover, in a typical GDT choice model, some consumer's behaviors may have a relatively high value of λ associated, so we may want to allow consumer's behaviors with high probabilities of occurrence to be split into more sub-behaviors. Hence, we also define a threshold τ that determines the maximum value of λ_k^q that we allow. We may express the number of sub-columns to generate as: $n_{min} - 1 + \lambda_k/\tau$

We typically take the values $n_{min} = 3, \tau = 0.01$, but this may depend on the number of products.

The number of consumer's behaviors in the final choice model is therefore less than $\frac{1}{\tau} + n_{min} * n_{col}$, where n_{col} scales with n (the number of products). In practice, this gives us 5-10 times the number of products, which allows the Mixed-Integer Optmization problem to be tractable while limiting the error.

3.4 Translating our solution to practical insights for the retailers

As we have seen, the approach presented in this chapter allows taking into account a high number of factors. Nevertheless, it may be not appropriate for certain applications, because of the hypothesis that we have formulated at the beginning. For example, if the price of some products changes significantly, the preference sequences of customers may change, and so may the choice model. In this section, we are going to show how it is possible to translate our choice models to practical insights for the retailers in the case where we do not want to apply all the process.

3.4.1 Matrix of strict preference

Given a Growing Decision Tree choice model, we can summarize the information of the rankings in a more concise way. Instead of considering the rankings of products, we can count, for all products i and j, the proportion of consumers that strictly prefer i to j. This information can be stored in a matrix B of size $n+1$, defined as:

$$B_{i,j} = \begin{cases} \sum_{k|\sigma_k(i)<\sigma_k(j)} \lambda_k & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{3.4}$$

In words, the component $B_{i,j}$ is equal to the proportion of consumers that prefer strictly the product i to the product j; and we ensure the diagonal terms to be equal to 0. We present in Figures 3.21 an example of a matrix of strict preference B for our small example of 6 products, where the trained GDT model is summarized in Figure 3.20.

We have, therefore, the property: $\forall(i,j), B_{i,j} + B_{j,i} \leq 1$. The cases of non-equality are when both i and j are in the indifferent products of at least one consumer's behavior (plus the diagonal terms). This matrix can be useful for helping the store manager to understand more easily his customers, and in particular for segmentation of clients, which is a topical issue in business intelligence.

3.4.2 Neighborhood

We define the **neighborhood** of an assortment as the set of assortments that have exactly one more or one less product. Using the GDT choice model, we can evaluate the change in revenue due to adding a product to the assortment. As explained in the Introduction (Chapter 1), adding a product has an impact on the revenue which is the algebraic sum of two effects:

- Increase in revenue due to the consumers that prefer the new product compared to the no-choice option
- Decrease of revenue due to the substitution of consumers from more profitable products to less profitable ones.

The Figure 3.22 illustrates this trade-off. Once the GDT choice model found, the revenues of all the assortments in the neighborhood can be computed very efficiently, therefore leading to recommendations.

3.4.3 Assortment optimization in practice: implication of the retailer in the decision

Even if the Mixed-Integer Optmization problem exhibited above allows to find a theoretically optimal assortment, we are conscious that our model may not catch some effects. Moreover, retailers may be reluctant to a tremendous change that would totally automatize the assortment decision process, which is nowadays a major responsibility for managers of retails stores. We are

Figure 3.20 Example of a GDT choice model

	0	0.2	0.2	0.2	0.2	0.2	0
	0	0	0	0	0	0	0
	0.5	0.5	0	0	0.5	0.5	0.5
	0.5	0.5	0.5	0	0.5	0.5	0.5
	0.3	0.3	0.3	0.3	0	0.3	0.3
	0.3	0.5	0.5	0.5	0.2	0	0.3
	0.5	0.5	0.5	0.5	0.2	0.2	0

Figure 3.21 Matrix of strict choice resulting from the GDT choice model

Figure 3.22 Consequences of adding a particular product to an assortment

going to present how the matrix of strict preference and the neighborhood analysis may be an efficient decision-support tool.

The retailer may, therefore, choose himself a base-assortment, as he would have done without the help of a software. Then, we may be able to run the process of training of choice model and integration of new products, resulting in a GDT choice model: $\{\sigma_k, \lambda_k\}$. It can be translated as a matrix of strict preference B as defined above. Then, this matrix can be used to segment its consumers, more efficiently than with the entire choice model (only n^2 scalars to compute).

The primary interest of the GDT choice model is that we know at which rank the information stops: it is after the preferred products (represented as a || in the consumer's behaviors). Hence, the matrix of strict preferences only takes into account real preferences; whereas if we had computed it for a fully-ordered choice model such as in Bertsimas and Mišic (2016), we would have included rankings that have no meaning in the training set.

Then, using the neighborhood defined in the previous subsection, we can compute the predicted revenue increase or decrease based on our choice model, and use this to suggest products to add or to remove to the assortment designed by the retailer.

As a conclusion, our approach is based on three main steps: efficient training of choice model, integration of new products, and assortment optimization. We have shown in this part the theoretical results on those three parts, as well as some insights on how to design a decision-support software for companies reluctant to set up the automatized whole process. In the next part, we are going to present the results validating our approach.

CHAPTER 4 THEORETICAL AND EXPERIMENTAL RESULTS

In the previous part (Chapter 3), we have presented the details of our solution in three main sections: training choice models, generalization to new products and assortment optimization. We are now going to to focus on the numerical results for each of these steps. Nevertheless, we begin this chapter by explaining the general process that we are going to follow to assess the models.

4.1 General process for assessing the models

In this section, we present our approach to assessing the models. Our models being data-driven, we are going to use two types of data: artificial and industrial. Artificial data is useful to quantify the ability of models to learn, compared to an optimal value that is known due to the nature of the choice model used to generate the data. Industrial data is noisy, sometimes contradictory, and we have to learn anyhow and see how our models perform in the presence of real-world data.

4.1.1 Data generation

The choice model used to generate the data is of paramount importance: we have to do it in a consistent way with the effects that we want to have in the data. Mainly, we want to be able to compare our results with those obtained in the paper of Bertsimas and Mišic (2016).

Utility based on the products: assessment of training

First, we want to assess the training part. For that, we will select a ground truth model able to generate profiles of consumers. The choice model mostly used for this purpose is the Multi-class MultiNomial Logit choice model. It assumes that T classes of consumers exist; we define a probability distribution among the classes: p_t (therefore, $\sum_t p_t = 1$ and $\forall\, t, p_t \geq 0$). Each class of consumer t gives each product i a utility $u_{t,i}$, and chooses the product i, from assortment S, with probability:

$$\mathbb{P}(i|S, \text{class}{=}t) = \frac{e^{u_{t,i}}}{e^{u_{t,0}} + \sum_{j \in S} e^{u_{t,j}}}$$

Therefore, the probability that a random customer chooses a product i is:

$$\mathbb{P}(i|S) = \sum_{t=1}^{T} p_t \frac{e^{u_{t,i}}}{e^{u_{t,0}} + \sum_{j \in S} e^{u_{t,j}}}$$

Randomly choosing the utilities For consistency with Bertsimas and Mišic (2016), we will generate the utilities the same way as in their paper (see p.21, section 5.3). Specifically, we generate a matrix q of the same dimension as u with the uniform law in $[0,1]$; we randomly select four products for each class t for which we pose $u_{t,i} = 10 * q_{t,i}$, and we set the $n-4$ other products' utilities to $u_{t,i} = 0.1 * q_{t,i}$.

Assortments As we have explained in the introduction, an assortment is a subset of the set of products. We, therefore, generate randomly two sets of M assortments, that will be the base of our training and test sets. Each assortment is a vector of binaries, and we impose that the density of the assortments is 0.5, which means that half of the products are present in each assortment. Unless specified differently, the number of assortments M in each set is fixed to 20.

Then, we can translate the utilities $u_{t,i}$ to a vector of sales probability of each product in each assortment: $v_{i,m} = \mathbb{P}(i|S_m)$. As previously, v is the vector of actual sales. Remember that we see it as a vector: each line is indexed by the tuple (i,m) so that we can see the matrix of choices $A^k_{i,m}$ as a matrix.

Therefore, the training set is composed of M assortments and the sales on them, encoded in the vector v, according to a ground truth choice model.

Test set After running the GDT algorithm or the one of Bertsimas and Mišic (2016), we compute the test error. For doing that, we have to compute the matrix of choices on the test set, A_{test}, and evaluate the test error defined as: $\sum_{i,m} |A_{\text{test}}\lambda - v|_{i,m}$.

Utility based on the features: assessment of the generalization

Next, we are going to present how to generate data in a consistent way for assessing the generalization to new products. In this part of the process, as explained in Chapter 3, we generalize a choice model to new products on which we have no transaction data, based on the features of both old and new products. Therefore, generating the data by using utilities of products directly would not be appropriate for this part, because this ground truth model would not take into account the characteristics of the products encoded in the features.

Hence, we have to adapt the ground truth choice model to use the features. We, therefore, first

choose the features of the products randomly: we choose 5 categorical features (that can represent for example the class of the product, the color, the material, the quality and the price band), and allow 10 possibilities for each feature. We end up with 50 different binary features, that we translate as a vector of 50 binary components. We call $b_{i,f}$ the binary value that is 1 if the product i has the feature f.

We set a vector of weights for the 50 features, to represent the different importance granted by consumers to each categorical feature. We decided to weight the 10 first components by a factor 10, the components $11-20$ by a factor 1, and the components $21-50$ by a factor 0.1:

$$\gamma_f = \begin{cases} 10 & \text{if } 1 \leq f \leq 10 \text{ (feature 1)} \\ 1 & \text{if } 11 \leq f \leq 20 \text{(feature 2)} \\ 0.1 & \text{if } 21 \leq f \leq 30 \text{ (feature 3)} \\ 0.1 & \text{if } 31 \leq f \leq 40 \text{ (feature 4)} \\ 0.1 & \text{if } 41 \leq f \leq 50 \text{ (feature 5)} \end{cases} \tag{4.1}$$

Then, we uniformly choose the utility q_f^t of each feature f for each type t of consumer, and define the utility of a class t of consumers for a product i as:

$$u_{t,i} = \sum_f \gamma_f q_f^t b_{i,f}$$

Once we have computed the utilities for each class and for each product, we can split the old and new products. We took a ratio of 70% of old products and 30% of new products.

In the training set, we only have access to data among the old products. Therefore, we generate M assortments **among the old products for the training set** and create the vector of actual sales v based on the old products' utilities.

We also generate M further different assortments **among all the products for the test set**, and generate the vector of actual sales v_{test} based on the utilities of all the products.

Then, the process described in Chapter 3 can be executed, with the steps of training choice model and generalization to new products. Once our choice model among the old and new products is computed, we can evaluate the test error among the test set, thus including old and new products.

4.1.2 Industrial data

In the previous section, we have presented how to generate data in a consistent way to assess the models. Nevertheless, our approach being data-driven, its interest in mainly to catch information from real industrial data, even if it is noisy.

Acknowledgments for data collection

JDA Labs is a research lab of JDA Software, a company providing software in multiple verticals such as supply chain management, retail planning, transportation, manufacturing and warehouse management. This data laboratory provided us with two data sets, which were anonymized for confidentiality purpose, such that it is impossible to retro-engineer it and find which real-life products correspond to which product identifier.

Processing the data

The first data set consists of sales of shoes and the second on sales of shirts. We will now refer to them as the *shoes data set* and the *shirts data set*. They both include assortment data, transaction data, features of all products, and characteristics of the stores. This four types of data were necessary for our approach, as we have seen in Chapter 3. We are going to present the preprocessing necessary to convert this data into the shape that is relevant to our model.

Definition of an assortment The assortment dataset consists of a CSV (*Comma Separated Values*) file, with information on the presence of each product at each day in each store. We define an assortment as a triplet (store, week, year): for a given assortment, we can extract the list of products present at this particular store at the week. Those products are, therefore, the options available to the customer visiting the store.

Dimensionality reduction of the transaction data The transaction data consists of a list of transactions, each being identified by the product id, the quantity sold, the time stamp and the store. With those two last values, we can link the transaction to an assortment that was defined before. Thus, we can convert the transaction data as a vector of the probability of selling a given product present in the assortment to a random customer: the vector of actual sales, v.

Defining the vector of features For the shoe dataset, we have a table with the features of each item, that are:

- Class: 5 categorical values

- Sub-Class: 12 unique values. There is not a direct correspondence such as "sub-class $\Longrightarrow$ class."
- Brand: 34 unique values
- Material: 16 unique values
- Color: 24 unique values
- Average price: continuous value

For the shirts data sets, we have those features:

- Class: 6 categorical values
- Color group: 12 categorical values
- Lifestyle: 2 categorical values (dress, casual)
- Pattern: 15 categorical values
- Fit: 2 categorical values
- Sleeve length: 2 categorical values
- Fashion: 3 categorical values
- Average price: continuous value

When the categorical variables have less possible values, and when the number of examples for each value is balanced (meaning that there are about the same number of examples of each label), the information conveyed is of better quality. See Annex B for some insights on Boltzmann's information theory.

In general, we notice that the features in the shirts data set were of better quality: there are more features with a few categorical values, which conveys *more* information.

Clustering the stores Our model needs a quite wide variety of assortments to deal with the substitution of products. The more assortments we use, the less probable is overfitting. On the other hand, the first hypothesis of this model is that the sales on all the considered assortments must have been produced by similar consumer's behaviors, therefore in the same places and at similar seasons. A good way of solving this trade-off is to design a clustering algorithm for stores on which we observe sales' similarities, and run the process separately on each cluster. We use some features of the stores:

- State and city (location)

- Climate(4 different categorical values)
- Price band low (30% cheapest sales), medium and high (30% most expensive sales)
- Percentage of sales in each sub-category

The most used algorithm for non-supervised clustering is the k-means algorithm, but it requires only continuous data (MacQueen, 1967). The features above mentioned are mixed: state, city and climate are categorical whereas the other are percentages of occurrence, therefore continuous. Huang has proposed a way of extending k-means Huang (1997) to mixed categorical and continuous data, which suits our case. The resulting algorithm is called the *k-prototypes*. We can tune the approximate number of stores being in each prototype with the hyper-parameter k. For most cases, Huang (1997)'s algorithm is sufficient. Nevertheless, when we want very low values of k, it is interesting to consider a newer version of the k-prototypes algorithm, proposed by Ahmad and Dey (2007), which allows centroïds to be more flexible for categorical data.

We have explained the general way of assessing the models, with generated and industrial data. Now, we are going to present the actual results that we obtained on our models.

4.2 Training choice models

As presented in the corresponding section of Chapter 3 (see section 3.1), the training phase that we propose consists of a Growing Decision Tree choice model, which is derived from the model of Bertsimas and Mišic (2016). Both models are what we call here ranking-based choice models, as they are based on rankings of products (that we call consumer's behaviors).

For the sake of being concise, we will refer to the model of column-generation with the local search heuristic presented in the paper of Bertsimas and Mišic (2016) as: **CG-LS**. Similarly, we will identify the algorithm of column generation with the growing decision tree choice model as: **CG-GDT**. We also still use the notation MNL to indicate the use of the Multinomial Logit choice model.

4.2.1 Proof of concept: Prediction accuracy is better than with parametric choice models

We have run the model on two data sets: the first one is composed by only 100 assortments, which we consider being the lower bound for allowing the model to get some substitution effect; and the second is on a much larger scale: 3565 assortments (which represents the sales of 229 stores during 18 weeks, totaling approximately 130 000 transactions).

The model CG-LS, as we have seen previously (see Section 3.1), trains by minimizing the objective value of the master problem, which is: $\sum_{i,m} |A\lambda - v|_{i,m}$. This is a L1 norm. In contrast, the

Multinomial Logit choice model minmizes the L2 error.

In general, if we name $p_{i,m}^{pred}$ the **predicted** probability of selling the product i in the assortment m, and $p_{i,m}^{obs}$ the **observed** probability of selling the product i in the assortment m, we can define the L1 and L2 errors as:

$$L1 = \sum_{(i,m)\in Test} |p_{i,m}^{pred} - p_{i,m}^{obs}|$$

$$L2 = \sum_{(i,m)\in Test} (p_{i,m}^{pred} - p_{i,m}^{obs})^2$$

We trained the MNL model and our implementation of CG-LS. For both instances (100 and 3565 assortments), we computed the $L1$ and $L2$ errors with both models, and summarized the results in Table 4.1. We can see that the CG-LS approach outperforms the MNL model not only with the L1 error (which was expected, since it is trained to minimize the L1 error), but also with the L2 error used to evaluate the differences.

Table 4.1 Ranking-based model outperforms standard MNL model with both L1 and L2 errors

Data	100 assortments		3565 assortments	
Model	BM	MNL	BM	MNL
L1	100.1	104.3	3672	3976
L2	2.55	2.68	97.3	101.4

Therefore, the ranking-based choice model is promising to be explored more. In the following, we are going to present the results in particular about the GDT choice model.

Definition of the error ϵ and the accuracy threshold ϵ_0

Bertsimas and Mišic (2016) use the Mean Absolute Error (MAE) as an indicator of the accuracy of prediction. It is defined as $\frac{|A\lambda - v|}{P}$, where P is equal to the sum of the number of products in all assortments. Nevertheless, our experiments show that the MAE as it is defined is a biased indicator of the accuracy of the algorithm when the number of products becomes large.

We have this supremum:

$$|A\lambda - v| \leq |A\lambda| + |v| = \sum_{i,m} (A\lambda)_{i,m} + \sum_{i,m} v_{i,m} = 2\sum_{m} 1 = 2M$$

The inequality is nearly saturated at the first iteration, because the components of both vectors are totally different when the prediction achieved by the vector of predicted sales $A\lambda$ is uncorrelated with the vector of actual sales v. Therefore, at first iteration: $|A\lambda - v| \approx 2M$.

We define the parameter ϵ that measures the ratio of decrease of the objective value of the master problem, between the first iteration and the current iteration. We use ϵ as a stopping criterion. We consider that near-optimality is achieved when: $|A\lambda - v| < 2M\epsilon$.

We can also find the relationship between ϵ and MAE when we reach the stopping criterion. We consider a sparsity of assortments of 0.5, which means: $P \approx 0.5Mn$ (remember that P is the sum of *the number of products in all assortments*). Then:

$$MAE = \frac{|A\lambda - v|}{P} \approx \frac{2M\epsilon}{0.5Mn} = 4\frac{\epsilon}{n}$$

Therefore, for high values of n (for example 1000), using as stop criterion $MAE = 10^{-3}$ as Bertsimas and Mišic (2016) do, leads to an abortion of the algorithm after the first iteration, while using $\epsilon_0 = 10^{-3}$ requires solving the program to near-optimality. ϵ can therefore really be understood as a ratio of decrease, whereas the MAE could be a misleading indication when considering large numbers of products.

We will eventually use ϵ to measure the error: we now call ϵ_0 the threshold that is used as stop criterion, and ϵ the value at an arbitrarily given iteration. For generated data, we will consider $\epsilon_0 \in \{10^{-1}, 10^{-2}, 10^{-3}\}$, and for real data we will consider higher ϵ_0, depending the computing time we allow.

We will call the train error and the test error respectively :

$$\epsilon_{tr} = \frac{|A_{tr}\lambda - v_{tr}|}{2M}, \epsilon_{te} = \frac{|A_{te}\lambda - v_{te}|}{2M}$$

Now, we are going to show practical results on the Growing Decision Tree algorithm CG-GDT, as compared to CG-LS.

4.2.2 Training and test error convergence

In this section, we plotted the training and test error in function of the time. Computationally, at each iteration, we stored the timestamp and the training and test errors.

Artificial data

As described previously (see section 4.1.1), we generate data for two instances of 10 and 100 products.

With 10 products (see Figure 4.1), we notice that the train error ϵ_{tr} is below 1% = 0.01 in less than 1 second with CG-GDT (in blue; or black when printed in grey-scale), while CG-LS takes 15

seconds to achieve this train error (in green; or light grey when printed in grey-scale).

The test errors (dot line) are in both case strictly decreasing, which means that the model is relatively resistant to overfitting, and always higher than the corresponding training errors. This is because we do not expect the accuracy to be better on the test set than on the training set.

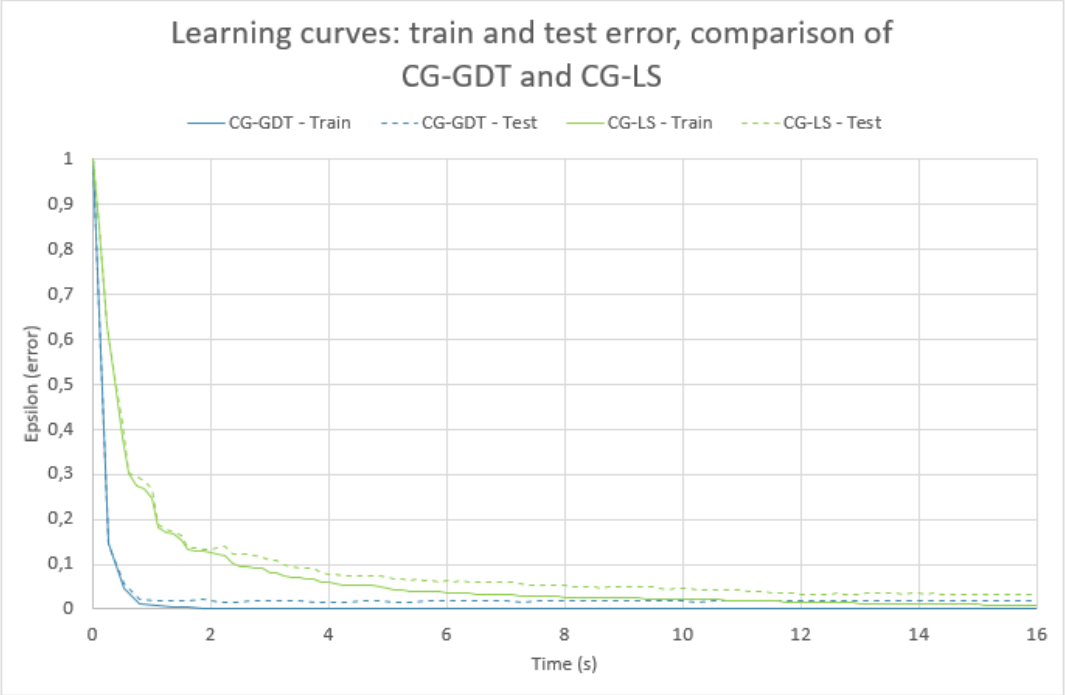

Figure 4.1 Learning curves for CG-GDT and CG-LS models, for 10 generated products

The instance with 100 products shows the same trends (see Figure 4.2), with a relative time improvement for CG-GDT, compared to CG-LS. Nevertheless, we notice that the model of CG-LS takes much more time to decrease its train error ϵ_{tr}. In particular, in the period observed (40 seconds), it is not able to decline below 37%, while the CG-GDT choice model achieves an error $\epsilon_{tr} = 1,5\%$ in 25 seconds.

We also notice that most of the iterations of CG-LS do not result in a significant decrease of the error; while some (seldom) iterations result in a significant reduction of the error. The iterations without decrease of the error may be understood as iterations where columns with a not small-enough reduced cost have been found with the local search heuristic. In contrast, the abrupt improvements in the training curve may be seen as iterations where the local search heuristic has found a critical column.

Example: why the first iterations of the CG-GDT lead to a huge decrease in error. We can consider a data set where the product 3 is crucial (we can assume that it has been purchased

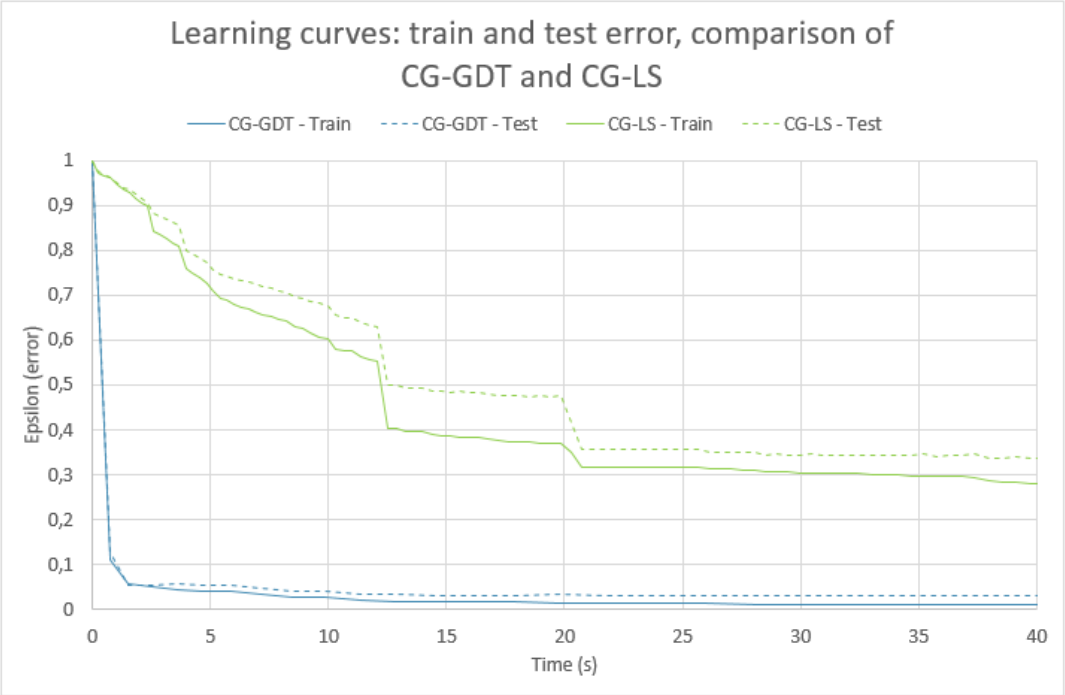

Figure 4.2 Learning curves for CG-GDT and CG-LS models, for 100 generated products

by a high fraction of the consumers when it was present); but in the assortments where the product 3 is not present, the product 17 is relatively well sold. For such a data set, a column with 3 in the first position, and 17 in second position $(3,17,...)$ is, therefore, necessary to learn the model properly. Nevertheless, it is entirely possible that CG-LS does not find a column such as $(3,17,...)$, and that the training error remains very high.

In contrast, with the CG-GDT, we generate at the beginning a column: $(3,...)$ (product 3 is the preferred, then indifference between the other products). Therefore, at the first iteration, the master problem can give significant weight to this column that is of paramount importance. When computing the reduced costs of the sub-behaviors of rank 1 of this column, the CG-GDT algorithm will notice that the column $(3,17,...)$ is associated with a very small reduced cost. This column will be selected, and at iteration 2, we will fully notice this effect of substitution of products 3 and 17.

Industrial data

We also conducted the same study on industrial data. We used the shoe data set described above (see Section 4.1.2), on a subset of 10 stores of the same cluster during 10 consecutive weeks, which gave us 100 assortments. There were 192 products in total. In Figure 4.3, we plotted the

training error for both methods, and we observe the same global trend.

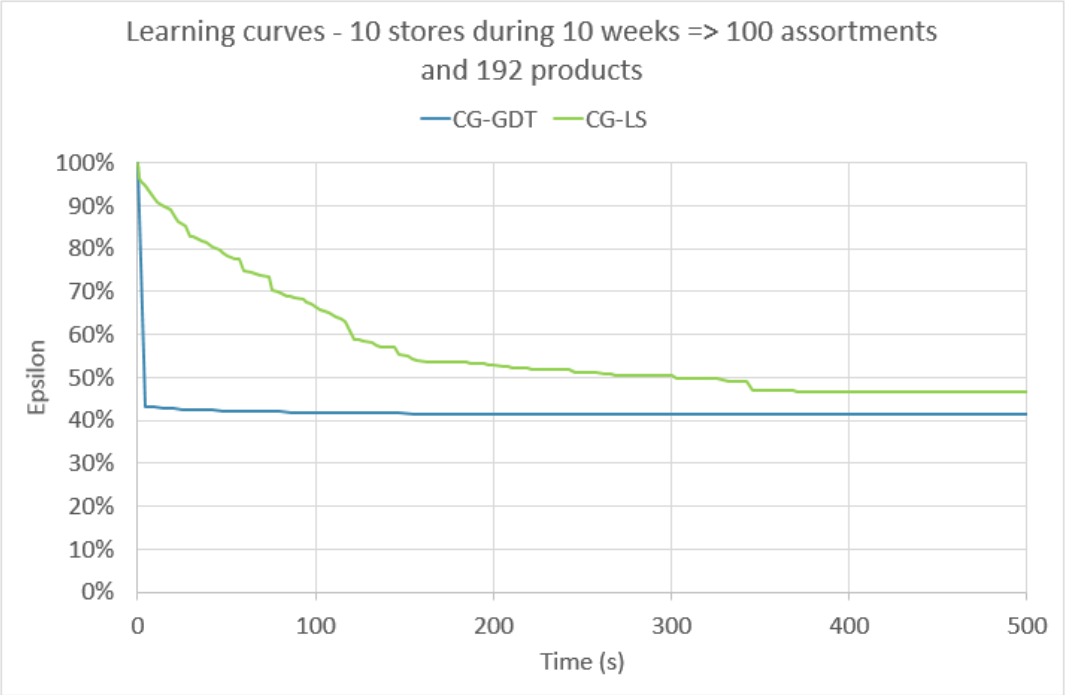

Figure 4.3 Learning curves for CG-GDT and CG-LS models, for 192 real products

Nevertheless, we notice that the optimal value is not 0 as it is the case for artificial data, because real data is noisy. We can converge to 41% with the GDT choice model, while Bertsimas and Mišic (2016)'s algorithm stops at 48% with the time limit of 500 seconds.

We are going to show a practical example to explain why real industrial data can be noisy. The main purpose of this thesis is to determine the impact of assortment on the buying behavior of the consumer. Nevertheless, we may notice two very similar assortments (we can assume for example: $S_1 = \{1,3,4,5\}$ and $S_2 = S_1 \cup \{7\}$). It may happen that sales are uncorrelated on those assortments: for example, the sales on S_1 are shared on products 1 and 3, and the sales on S_2 are on products 4 and 5. In such a case, it is not possible to find a ranking-based choice model that fits both those assortments. Hence, such sales data would result with non-null optimal value.

We notice that the CG-GDT performs very well since the first iteration even for industrial data because it can decrease the error to near optimality in one iteration.

Hence, our CG-GDT outperforms GD-LS regarding computation times both for learning artificial and industrial data, and it can be efficiently generalized to a test set.

4.2.3 Experimental evaluation of the complexity vs. other

In this subsection, we evaluate experimentally the complexities of both algorithms: the baseline is CG-LS and our new way of evaluating the choice model in the shape of a Growing Decision Tree: CG-GDT. We consider three parameters: the number of products n, the number of assortments M, and the accuracy threshold ϵ_0. We will study this around the "standard point" defined by $n = 10$, $M = 20$, $\epsilon_0 = 0.1$. Each point in the graphs plotted will be the averaged value over 30 instances, to limit the impact of randomization.

Accuracy threshold ϵ_0

We set the number of assortments to 20 and the number of products to 10, and we vary the accuracy threshold $\epsilon_0 \in \{10^{-1}, 10^{-2}, 10^{-3}\}$. We notice that both computing times are on a line, which may suggest that the complexity is logarithmic (because the scale is logarithmic), with a regression coefficient $R^2 > 0.98$.

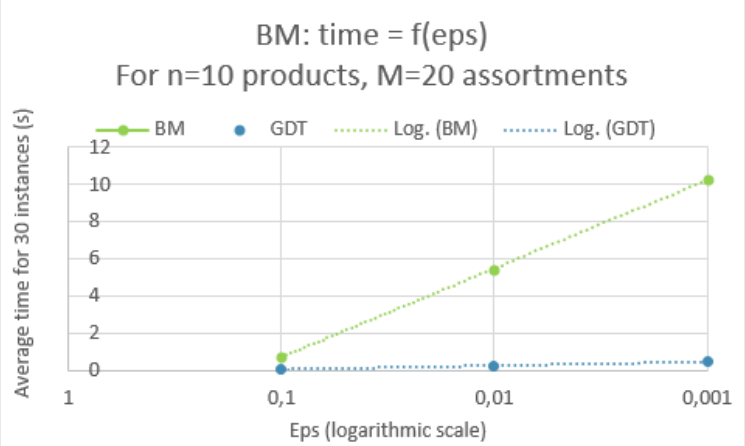

Figure 4.4 Comparison of the complexity for both models: impact of ϵ_0

Number of assortments M

We set the number of products to 10, and ϵ_0 to 0.1, and we vary the number of assortments $M \in \{10, 20, 50, 100\}$. We notice a linear complexity of both algorithms, with $R^2 > 0.99$.

Number of products n

We set the number of assortments to 20 and ϵ_0 to 0.1, and vary the number of products $n \in \{10, 20, 50, 100\}$. We notice a linear complexity for the GDT algorithm ($R^2 = 0.99$), but the BM

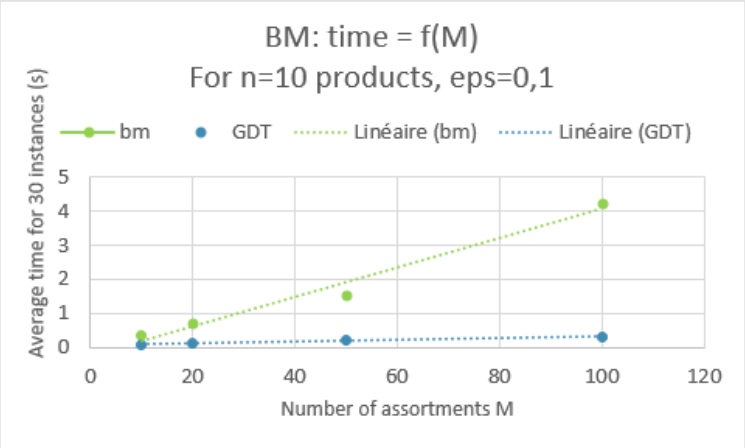

Figure 4.5 Comparison of the complexity for both models: impact of M

algorithm suggests a cubic complexity ($R^2 = 0.998$).

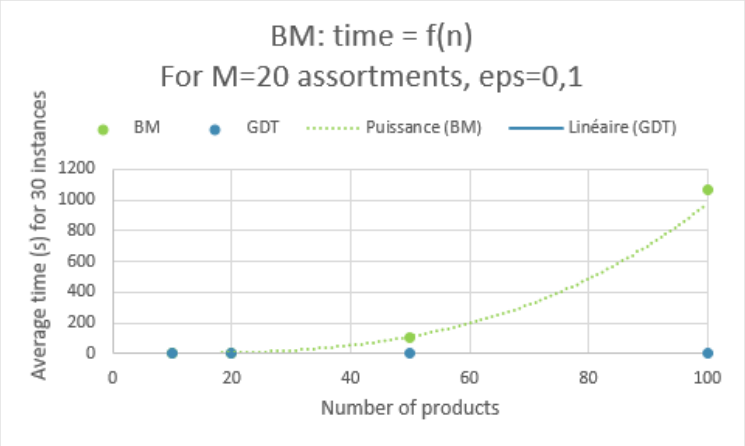

Figure 4.6 Comparison of the complexity for both models: impact of n

Observed computational complexity

It would be unfair to express the worst-case computational complexity: indeed, CG-LS is based on a random search of a new column at each iteration, which means that it may take, in the worst case, $n!$ steps to find a negative reduced cost new column with CG-LS. Nevertheless, we may compare the observed computational complexity of both alforithms, with both methods, around the standard-point $n = 10$, $M = 20$, $\epsilon_0 = 0.1$.

When we compile the previous results, we can express the observed computational complexity of both algorithms in function of m, n, ϵ_0. The observed complexity of the GDT is therefore

$O(nM\ln(\epsilon_0))$, while the observed complexity of the CG-LS approach is: $O(n^3 M\ln(\epsilon_0))$. This explains why our method scales better with high numbers of products. We have been able to solve problems up to 1000 products while the paper of Bertsimas and Mišic (2016) does not explore what happens for more than 30 products.

4.3 Whole process on generated data

In this section, we are going to present the results we have achieved on the generalization to new products and assortment optimization.

4.3.1 Expected revenue vs. Ground truth revenue

To generate the artificial data, as explained above (see Subsection 4.1.1), we have used a choice model. This is called the ground truth model (**GT**), because it represents the assumed reality. We use the Multi-class MultiNomial Logit choice model, and we are therefore able to compute the revenue given by an arbitrary assortment: indeed, with the ground truth model, we know the assumed behavior of our customers, and we can therefore predict their behavior on a given assortment (given this model).

This computation is relatively quick, which allows us to compute the revenues of all possible assortments when the number of products is not too high. In this case, we can easily find the optimal assortment with explicit enumeration.

Nevertheless, when the number of products becomes too high, the exponential number (to be precise, 2^n) of possible assortments makes it intractable to explicitly compute all the revenues of assortments. Thus, we can still use a simulation approach that computes revenues of as many assortments as possible, and select the best among them.

When we run our entire process (training, generalization to new products, and assortment optimization), we end up with two major outputs:

- A GDT choice model, consisting of a list of tuples (σ_k, λ_k)
- The optimal assortment S^* according to this GDT choice model

The assortment S^* achieves the provably best revenue among the 2^n possible assortments, given the GDT choice model. Nevertheless, its value in the ground truth choice model may be different, in particular when we have not trained the GDT choice model to optimality (for example, for an accuracy threshold ϵ_0 different than 0). Therefore, we have to compare the revenue of S^*

when computed with the GDT choice model ($R_{GDT}(S^*)$) to its revenue when computed with the ground truth model ($R_{GT}(S^*)$).

Therefore, it is possible (and it happens in practice) that the optimal assortment according to the GDT choice model is different from the optimal assortment according to the ground truth model. To ensure a coherent computation, we therefore compare the revenues computed with the ground truth model.

4.3.2 Expected revenue increase: brute force vs. our solution

In this subsection, we compare our approach with brute force, i.e. an approach based on explicit enumeration.

How often does the algorithm find the actual optimal assortment despite the limited data?

We are in a context of limited data, because we generate a small number of assortments ($M = 20$ assortments is much smaller than the total number of assortments, 2^n for $n \geq 10$).

We found that for up to 19 products, it is possible to evaluate in reasonable computing time the revenues for the ground truth model of all the possible assortments. Indeed, there are $2^{19} = 524,288$ different assortments for $n = 19$, which is computable. We, therefore, compared both assortments: S^*_{GDT} (optimal assortment with the GDT choice model) and S^*_{GT} (optimal assortment with the Ground Truth choice model with brute force). Figure 4.7 plots the ratio of instances where we found the actual optimal assortment ($S^*_{GDT} = S^*_{GT}$).

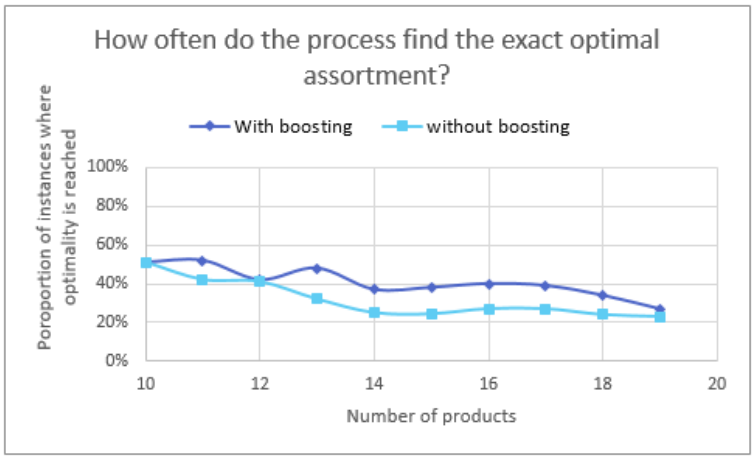

Figure 4.7 Proportion of instances for which optimality is reached with and without boosting

Boosting

We also compared the effect of the extension of the GDT choice model to the model of Bertsimas and Mišic (2016), that we described in Section 3.3.2. It is possible to translate the GDT choice model with the heuristic that consists in expanding each leaf of the GDT tree to a random fully-ordered permutation. Nevertheless, as we pointed out, we may also boost the results in tuning the parameters (n_{min}, τ), which give the number of sub-columns to generate for each leaf (see Subsection 3.3.2 for the definitions of those parameters). Let us remind that the number of sub-columns to generate is: $n_{min} - 1 + \lambda_k/\tau$

We chose the values of these parameters: $n_{min} = 3, \tau = 0.01$ in the instances called *With boosting*; and the values $n_{min} = 1, \tau = 1$ in the instances called *Without boosting*. We show in Chart 4.2 the number of sub-behaviors that are going to be generated for those values of the parameters, for a GDT instance with 10 products and 12 columns, of probabilities λ_k, for $k \in [\![1, 12]\!]$.

In Table 4.2, we see that for those values of the parameters $n_{min} = 3, \tau = 0.01$, we have before the assortment optimization problem a choice model with 10 times more columns (124 vs. 12). This results in more computation time for the assortment optimization problem, but provides much better results.

This technique is often used to prevent models from overfitting. For example, Bertsimas and Mišic (2016) propose to train several times the choice model, and average the results. It is mush faster to tune the number of sub-columns to generate with the value of the probability associated to each column with a CG-GDT model, rather than training several times a CG-LS choice model.

Indeed, we see in Figure 4.7 that we find the actual exact assortment ($S^*_{GDT} = S^*_{GT}$) more often with the boosted GDT model. For example, with 16 products, for 40% of the instances instead of for 24% of the instances without boosting.

Optimality gap when the optimal assortment is computable by brute force

As we have seen with Figure 4.7, in 60% to 70% of the instances, the process did not find the exact assortment achieving the real lowest value when its revenue is evaluated with the ground truth model (even if reaching the optimal value of the assortment optimization problem). We denote R_{GT} as the function associating an assortment S to its revenue predicted with the Ground Truth model. We may, therefore, interest ourselves to the optimality gap, which we define as:

$$\text{Optimality Gap} = \frac{R_{GT}(S^*_{GT}) - R_{GT}(S^*_{GDT})}{R_{GT}(S^*_{GT})}$$

Table 4.2 Number of sub-behaviors in the final choice model without and with boosting

k	λ_k	Without boosting	With boosting
1	20%	1	22
2	15%	1	17
3	12%	1	14
4	10%	1	12
5	10%	1	12
6	10%	1	12
7	7%	1	9
8	6%	1	8
9	5%	1	7
10	3%	1	5
11	1%	1	3
12	1%	1	3
Sum	100%	12	124

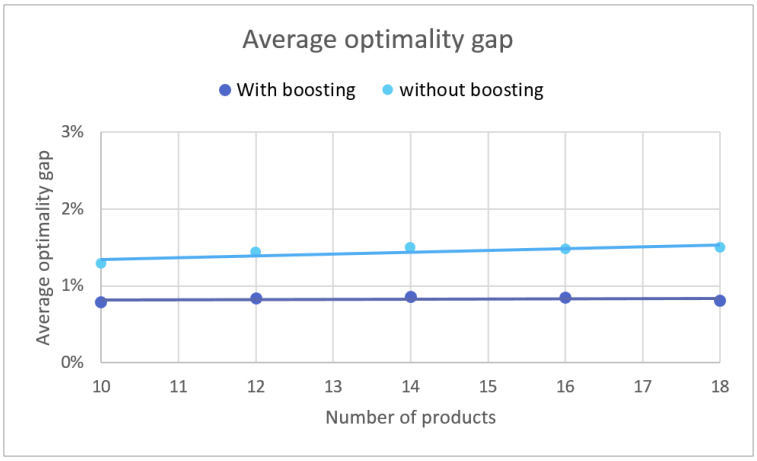

Figure 4.8 Average optimality gaps of assortments with and without boosting

We plotted the average optimality gap over 30 instances for the number of products $n \in \{10, 12, 14, 16, 18\}$. We did that once for both methods: with and without boosting. The results are shown in Figure 4.8.

We notice that the optimality gap is in average always smaller than 1.6%, slightly increasing with the number of products. We also note that the boosting method constantly provides lower optimality gaps.

Simulation gap when optimal value is not computable

When considering higher values of the number of products, it is not tractable anymore to find the optimal assortment per explicit enumeration (for example, one billion years of computation time would be necessary to evaluate all assortments for an instance with $n = 60$ products). Therefore, we have to find another approach to assess the optimized assortment.

We run our entire process (training, generalization, and optimization), which gives us the optimized assortment S^*_{GDT} according to the GDT choice model. We measure the time required by the entire process, t_{GDT}. Then, we run a simulation procedure that randomly evaluates the most probable assortments and their revenues, with t_{GDT} as the time limit. Hence, both approaches use the same time limit.

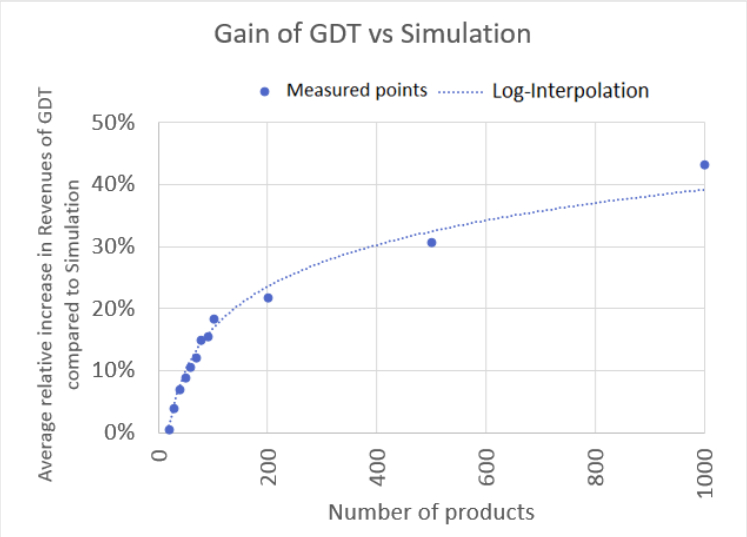

Figure 4.9 Average gain of CG-GDT versus simulation, for high values of n

The GDT choice model results in an optimal assortment S^*_{GDT} and an associated revenue (evaluated with the ground truth model) of $R_{GT}(S^*_{GDT})$. The simulation approaches returns, in the limit of time described above, the assortment S_{GT} with the highest found revenue $R_{GT}(S_{GT})$. We removed the *star* in the notation of the assortment S_{GT}, because it is not provable optimal (since we have not been able to check for all assortments). Hence, we can define the simulation gap as:

$$\text{Simulation Gap} = \frac{R_{GT}(S_{GT}) - R_{GT}(S^*_{GDT})}{R_{GT}(S_{GT})}$$

Then, we consider the numbers of products $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 500, 1000\}$, and run for each of them 30 instances where we compare the simulation gap as defined above.

We plotted in Figure 4.9 the results showing the simulation gap in function of the number of products.

It appears that our approach outperforms the simulation as soon as the time allowed for the simulation is small enough for all assortments not to be computed; when the number of products increases, the difference becomes even more striking. Intuitively, we can deduce that the assortments achieving a close to optimality revenue are seldom in the space of all assortments.

Therefore, we have seen in this section that our approach allows finding efficiently quasi-optimal assortments, that outperform the brute force approach. We can also note that CG-LS algorithm would not have been able to converge for big values of the number of products n, as stated previously.

4.4 Industrial data: impact of capacity constraint on expected revenue increase

In practice, retailers have not the possibility to expose a lot of products in a shelf; but they can choose their small assortment from a catalog of several hundreds of products. In this section, we are going to show how to handle capacity constraints, and how it affects the solution and the computation times.

As demonstrated previously (see Subsection 2.2.4), the mixed-integer program of assortment optimization is able to deal with capacity constraint. To set an upper bound U to the number of products inserted in an assortment, we can add to the problem the following constraint:

$$\sum_{i=1}^{n} x_i \leq U$$

For the shirts data set, we selected 7 stores (in the same cluster of stores) and their sales for the months of April to July 2015. We aggregated the assortments at the day level, because there were much more sales in this data set than in the shoes data set, and split the assortments between the training set (50%) and test set (50%). We excluded the products that were sold less than 10 times in all the data set, considering that we had not enough data to predict something among them. We defined the new products as the products that were present in the test set but not in the training set.

We had 161 old products and 35 new products, thus 196 products in total. First, we trained our GDT choice model to near-optimality, then we applied the process of generalization to new products. Then, we solved the assortment optimization problem without capacity constraint. The optimal solution was found to have 91 products (over the total of 196 products).

Our way of assessing the result is different when using the artificial data: indeed, since we deal with real data, we have no access to the ground truth model (if it was the case, then this entire thesis would not be of interest...!), able to specify the revenue of an assortment. Instead, we use the test set and evaluate the revenues of all the assortments in the test set based on the GDT choice model that we have trained. We select the assortment S^*_{test} that achieves the best revenue according to the GDT choice model on the test set $\mathbb{T}$:

$$S^*_{test} = \text{argmax}_{S_m \in \mathbb{T}} R_{GDT}(S_m)$$

We call this revenue $R_{GDT}(S^*_{test})$ the baseline. Then, we compute the relative increase of revenue (*Rev. Incr.*) predicted by the GDT model when we compare the best assortment S^*_{test} to the optimal assortment returned by our entire process: S^*_{GDT}.

$$\textit{Rev. Incr.} = \frac{R_{GDT}(S^*_{GDT}) - R_{GDT}(S^*_{test})}{R_{GDT}(S^*_{test})}$$

We then solved the assortment optimization problem with the capacity constraint, where the upper bound is set to the values: $U \in \{25, 40, 50, 65, 100\}$. Naturally, with $U = 100$, the constraint has not been saturated (and we have obtained the same optimal assortment as without the capacity constraint).

For lower values of U, we observed that for each of them the capacity constraint was saturated; i.e., the number of products in the optimal assortment has been in practice equal to the upper bound that we set. We plotted in Figure 4.10 the Revenue Increase defined above, in function of the capacity constraint. We observe that the predicted increase in revenue can be as high as 45% for the relaxed problem, but that even small stores able to carry as little as 25 products may benefit from 35% of revenue increase.

4.4.1 Computing times for the whole process

In this Section, we are going to explore the computing times for executing each step of our process. All our computational experiments of this part were done on the machine Rossoblu: Intel Xeon E5-2637, 9.6 GT/s, 4C/8T.

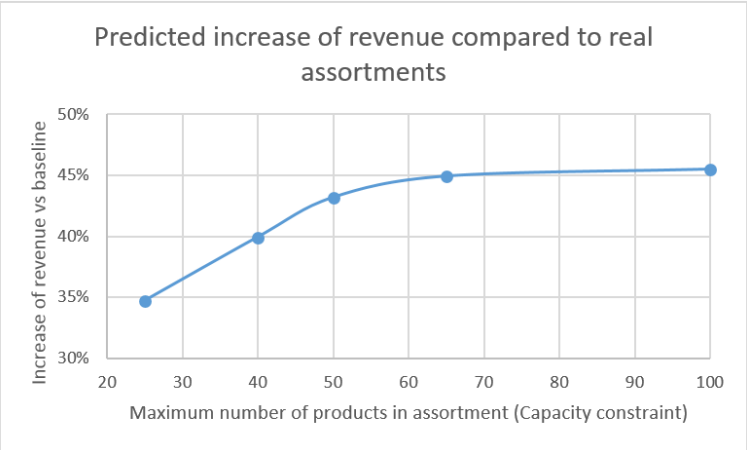

Figure 4.10 Impact of the capacity constraint on the predicted revenue increase

Training

The training of choice models with the GDT choice model depends on the accuracy threshold ϵ_0, the number of products n and the number of assortments M. As we have seen, the GDT choice model can learn about $80-85\%$ of all the information in only one iteration: therefore, if ϵ_0 is not sufficiently small, then the training is going to be stopped in only one iteration. This would result in a very simple GDT choice model because there are only one product ranked in each consumer's behavior after this first iteration. At the stage of the first iteration, we have not used the main interest of the GDT choice model, which is to understand the substitution behavior of consumers. Therefore, it is not appropriate to stop the learning phase at the first iteration.

We plotted in Figure 4.11 the computation time in function of the number of iterations, for an example of $n = 161$ products. We must note that in most cases, only a few dozens of iterations are necessary to understand most of the substitution patterns in a given dataset: for example, 50 iterations are done within 20 minutes, which is a reasonable computation time for such an instance.

Generalization

The generalization to new products has not been the bottleneck when we applied the whole process: it does not require to solve any mathematical problem, and consists only in the computation of scalar products. In practice, it takes always less than 10% of the time required for the training step.

Nevertheless, depending on the parameters of the generalization, we may end this step with a huge choice model, which is going to be a drawback for the assortment optimization step.

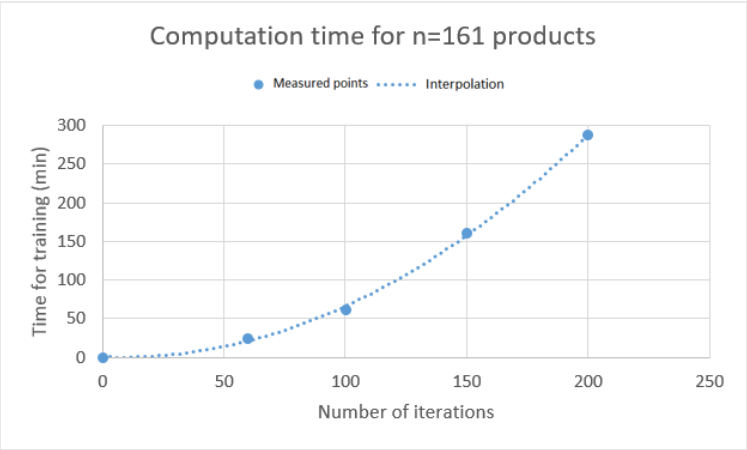

Figure 4.11 Computation time in function of the number of iterations, for $n = 161$ products

Indeed, the generalized choice model for an instance with 161 old products and 35 new products results in a choice model with eight times more columns than the number of columns after the training phase.

Assortment optimization

The assortment optimization computation times depend on the number of products n and on the size of the choice model provided to it (the number of columns K). In practice, if we have not to generalize then the choice model is of a small enough size (most of the times, $K < n$ for near-optimality). Nevertheless, if a generalization phase is inserted in the process to deal with new products, then the computation times may be higher.

Computation times

We launched a batch of 30 instances for each of the values of the number of products $n \in \{10, 50, 100, 200\}$ the computation times required to train the choice model and run the assortment optimization problem. In Table 4.3, we summarized the average time required for each condition, and the standard deviation observed.

We also launched instances for high numbers of products for two tasks:

- Training and assortment optimization (without new products)
- Training, generalization to new products and assortment optimization

We summarized in Table 4.4 the highest numbers of products for which we were able to realize

Table 4.3 Average (and standard deviation) of the time required to train and optimize

n	**Training the CM (s)**	**Assortment Optimization (s)**
10	0.382 (+/- 0.040)	0.0676 (+/- 0.0206)
50	32,6 (+/-1.89)	7,17 (+/- 3.04)
100	399 (+/- 19.0)	41,0 (+/- 16.1)
200	828 (+/-42.6)	270 (+/-102)

the tasks above within several time limits t, for $t \in \{1\text{sec}, 1\text{min}, 1\text{hour}, 1\text{day}\}$. Those limits are the different orders of magnitude that we may be interested in; 1 day consists in the maximum time for industrial application, according to our industrial partners.

Table 4.4 Maximum number of products for achieving a task within a time limit

Time limit:	**1 sec**	**1 min**	**1 hour**	**1 day**
Train & Asst optimization	20	70	400	1000
Train, Generalization & Asst optimization	10	50	150	400

CHAPTER 5 CONCLUSION

5.1 Synthesis of our work

The assortment optimization problem is of paramount importance in practice for retailers, as well as an academically interesting challenge for the mathematical optimization and data science communities. In this thesis, we have presented an efficient data-driven process to go from raw transaction data to assortment optimization.

Extracting useful information from a transaction data set is not trivial, and the way of modeling the choice of consumers strongly influences the outcome. The hypothesis made at this stage are fundamental, as they limit the range of abstraction that our model may reach. In particular, in the context of assortment optimization, we wanted to extract the impact of assortment on the customer behavior. We, therefore, focused on ranking-based choice models, that group consumers depending on their supposed preference sequence among the products. The structure of such choice models provides the advantage of being easily exploited in a mixed-integer optimization program, that scales well with the number of products due to its limited number of non-continuous variables.

However, the training of such a choice model may be excessively slow when considering complex substitution effects between products and high numbers of products. The state of the art method for doing this is based on a column generation approach, which identifies potential columns by a local search heuristic in a factorialy big space. We, therefore, proposed to structure the space in which the local search operated to find new columns more efficiently. Moreover, we modified the choice model in allowing indifference between products at a certain rank. This is based on the idea that a substantial computational effort was spent on tuning a large number of ranks which are meaningless in practice.

We first validated our approach with artificially generated data, on which we compared two parameters: the time required to solve the problem, and the maximum number of products for which we were able to solve the problem. In both cases, we outperformed the state of the art by one order of magnitude. We also used industrial data from two major US retailers, and showed that the proposed algorithms ran equally fast on artificial and industrial data.

Moreover, the problem of assortment optimization can be extended to new products, on which we have no transaction information. For example, in a season-to-season prediction of assortments, retailers have to add products of the new season in the assortments. We, therefore, proposed an extension of the work to include new products in the set of products. The only data

on which we have access to are the features in common between old and new products. We proceeded with a likelihood approach, learning to generalize the behavior of consumers based on their features. The main idea was to state that among the consumers behaving in a certain way among the old products, then we may witness several behaviors among the new products. Finally, we presented how to practically solve the assortment optimization problem with this new choice model, even if new products have been added.

We conducted numerical studies to check the overall process. In particular, we compared the revenue of the assortments of the test set to the predicted revenue of the optimized assortment. We have seen that the process systematically led to larger assortments than what observed in the data set, with more products. This can be understood as the importance of offering a wide choice to consumers to increase the conversion rate. To take operational requirements into account, such as limited shelf size, we have inserted in the assortment optimization a capacity constraint, limiting the number of products in the optimized assortment to what is observed in average in the test set. The results show that even smaller assortments leads to a predicted 35% increase in revenue.

5.2 Future reaserch directions

The main goal of this work was to study assortment optimization, and therefore the impact of the assortment on the buying decisions of customers. The ranking-based choice models, which are interesting for the structure that they allow to provide to the assortment optimization problem, have nevertheless some limitations.

In particular, they assume an average behavior among all the training set, that will be transposed to all predictions. If some key-drivers must change from the training to the test set, such as demographic parameters, then the model may be inappropriate. In our work, we assume that the characteristics of the sales do not change. We were able to deal with this problem with the clustering of the stores that we presented.

Moreover, as stated in the introduction, we consider that the prices of products are fixed over the time. In some retail applications, this hypothesis limits the ranges of weeks during which we may use our approach. Indeed, the clothing industry is known to be driven by huge promotions and incentives to the customer, that we may not be able to take into account. To take this into account, we separated the study when prices changed.

The generalization consists in providing forecasts for a fixed set of new products, described by their features. Our way of dealing with this problem increases by a lot the number of consumer's behaviors: this may be problematic when considering too many new products. Moreover, its

generalization accuracy stays acceptable only when the number of old products remains higher than the number of new products. Nevertheless, something better for the retailer would be to design from scratch the new products that would be interesting to add to the assortment. This is known as the product line design problem. We could see this problem as an optimization problem derived from the assortment optimization mixed-integer problem where we would also choose the features of a certain number of products to be added to the assortment. We could also design this problem as a two-stages optimization problem, where we first solve the assortment optimization problem among the old products, and then we design new products.

Moreover, instead of doing the whole process computationally (training of choice model, insertion of new products, assortment optimization), we may let the manager take the responsibility of the process of assortment decision, while providing him with insights derived from our model. We believe that helping the manager with data-driven tools is the best way of implementing a paradigm shift in a retail company: the managers may progressively increase their confidence in the recommendations of the tool, and thus will be willing to delegate an increasing part of responsibility to the automated process. Further, if the automated software makes errors due to the hypothesis of modeling, then the software company may improve the process with limited damage for the retailer.

REFERENCES

A. Ahmad and L. Dey, "A k-mean clustering algorithm for mixed numeric and categorical data," *Data & Knowledge Engineering*, vol. 63, no. 2, pp. 503–527, 2007.

K. J. Arrow, *Social Choice and Individual Values*. John Wiley & Sons, 1951, no. 12.

M. E. Ben-Akiva and S. R. Lerman, *Discrete choice analysis: theory and application to travel demand*. MIT press, 1985, vol. 9.

D. Bertsimas and V. V. Mišic, "Data-driven assortment optimization," *submitted to Management Science*, 2016.

T. F. Bresnahan, S. Stern, and M. Trajtenberg, "Market segmentation and the sources of rents from innovation: Personal computers in the late 1980's," National bureau of economic research, Tech. Rep., 1996.

J. J. M. Bront, I. Méndez-Díaz, and G. Vulcano, "A column generation algorithm for choice-based network revenue management," *Operations Research*, vol. 57, no. 3, pp. 769–784, 2009.

G. P. Cachon, C. Terwiesch, and Y. Xu, "Retail assortment planning in the presence of consumer search," *Manufacturing & Service Operations Management*, vol. 7, no. 4, pp. 330–346, 2005.

C. Chu, "A paired combinatorial logit model for travel demand analysis," in *Transport Policy, Management & Technology Towards 2001: Selected Proceedings of the Fifth World Conference on Transport Research*, vol. 4, 1989.

J. M. Davis, G. Gallego, and H. Topaloglu, "Assortment optimization under variants of the nested logit model," *Operations Research*, vol. 62, no. 2, pp. 250–273, 2014.

F. Y. Edgeworth, "The Mathematical Theory of Banking," *Journal of the Royal Statistical Society*, vol. 51, no. 1, pp. 113–127, 1888.

V. F. Farias, S. Jagabathula, and D. Shah, "A nonparametric approach to modeling choice with limited data," *Management Science*, vol. 59, no. 2, pp. 305–322, 2013.

K. J. Ferreira and J. Goh, "Assortment Rotation and the Value of Concealment," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper, 2016.

V. Gaur and D. Honhon, "Assortment planning and inventory decisions under a locational choice model," *Management Science*, vol. 52, no. 10, pp. 1528–1543, 2006.

P. M. Guadagni and J. D. Little, "A logit model of brand choice calibrated on scanner data," *Marketing science*, vol. 2, no. 3, pp. 203–238, 1983.

D. M. Hausman, *Preference, value, choice, and welfare*. Cambridge University Press, 2011.

H. Hotelling, "Stability in competition," *The Economic Journal*, pp. 41–57, 1929.

Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *In The First Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1997, pp. 21–34.

S. Jagabathula, "Nonparametric choice modeling : applications to operations management," Thesis, Massachusetts Institute of Technology, 2011.

——, "Assortment Optimization Under General Choice," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2512831, 2014.

A. G. Kök, M. L. Fisher, and R. Vaidyanathan, "Assortment planning: Review of literature and industry practice," in *Retail supply chain management*. Springer, 2008, pp. 99–153.

K. J. Lancaster, "A New Approach to Consumer Theory," *Journal of Political Economy*, vol. 74, no. 2, pp. 132–157, 1966.

K. Lee, K. C. Kang, and J. Lee, "Concepts and guidelines of feature modeling for product line software engineering," in *International Conference on Software Reuse*. Springer, 2002, pp. 62–77.

J. MacQueen, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967.

S. Mahajan and G. Van Ryzin, "Stocking retail assortments under dynamic consumer substitution," *Operations Research*, vol. 49, no. 3, pp. 334–351, 2001.

S. Mahajan and G. J. van Ryzin, "Retail inventories and consumer choice," in *Quantitative Models for Supply Chain Management*. Springer, 1999, pp. 491–551.

D. McFadden, "Conditional logit analysis of qualitative choice behavior," in *Frontiers in Econometrics*, P. Zarembka, Ed. New York: Academic Press, 1973.

——, "Modeling the choice of residential location," *Transportation Research Record*, no. 673, 1978.

——, "Econometric models for probabilistic choice among products," *Journal of Business*, pp. S13–S29, 1980.

K. S. Moorthy, "Market segmentation, self-selection, and product line design," *Marketing Science*, vol. 3, no. 4, pp. 288–307, 1984.

M. Mussa and S. Rosen, "Monopoly and product quality," *Journal of Economic theory*, vol. 18, no. 2, pp. 301–317, 1978.

P. Ray, "Independence of Irrelevant Alternatives," *Econometrica*, vol. 41, no. 5, pp. 987–991, 1973.

P. Rusmevichientong, B. Van Roy, and P. W. Glynn, "A nonparametric approach to multiproduct pricing," *Operations Research*, vol. 54, no. 1, pp. 82–98, 2006.

P. Rusmevichientong, Z.-J. M. Shen, and D. B. Shmoys, "Dynamic assortment optimization with a multinomial logit choice model and capacity constraint," *Operations research*, vol. 58, no. 6, pp. 1666–1680, 2010.

P. Rusmevichientong, D. Shmoys, C. Tong, and H. Topaloglu, "Assortment Optimization under the Multinomial Logit Model with Random Choice Parameters," *Production and Operations Management*, vol. 23, no. 11, pp. 2023–2039, 2014.

A. Sen, A. Atamturk, and P. Kaminsky, "A conic integer programming approach to constrained assortment optimization under the mixed multinomial logit model," IEOR, University of California, Berkeley, University of California, Berkeley, CA 94720–1777, Research report BCOL.15.06, 2015.

A. Sen, "The impossibility of a Paretian liberal," *Journal of political economy*, vol. 78, no. 1, pp. 152–157, 1970.

S. A. Smith and N. Agrawal, "Management of multi-item retail inventory systems with demand substitution," *Operations Research*, vol. 48, no. 1, pp. 50–64, 2000.

C. S. Tang, K. Rajaram, A. Alptekinoğlu, and J. Ou, "The benefits of advance booking discount programs: Model and analysis," *Management Science*, vol. 50, no. 4, pp. 465–478, 2004.

K. E. Train, *Discrete choice methods with simulation*. Cambridge university press, 2009.

C.-H. Wen and F. S. Koppelman, "The generalized nested logit model," *Transportation Research Part B: Methodological*, vol. 35, no. 7, pp. 627–641, 2001.

H. C. Williams, "On the formation of travel demand models and economic evaluation measures of user benefit," *Environment and planning A*, vol. 9, no. 3, pp. 285–344, 1977.

APPENDIX A NUMBER OF COHERENT CONSUMER'S BEHAVIORS

We consider a consumer's behavior on the old products, σ, and compute the number of behaviors over both old and new products that are coherent with it, C (*i.e.* behaviors that preserve the order of preference among the old products). This is useful in Subsection 3.2.3.

- n_r: number of old products ranked in the consumer's behavior
- n_{new}: number of new products
- $\mathfrak{C}_\sigma = \{\sigma_{\text{all}}/\sigma_{\text{all}}$ coherent with $\sigma\}$
- $C = |\mathfrak{C}_{sigma}]$ the cardinal of the set $\mathfrak{C}_\sigma$

The problem is that not all of the new products may be ranked in a new coherent behavior. Let's compute the number of coherent behaviors where k new products have been ranked. We will illustrate the calculation with the example of 3 old products (1,2 and 3) and 3 new products (a,b and c).

- $k = 0$

  → only 1 behavior: σ

  1 2 3

- $k = 1$

  → $(n_r + 1) * n_{new}$ behaviors because there are $(n_r + 1)$ possible ranks for the new product and n_{new} ways to choose it.

  x 1 2 3 1 2 x 3

  1 x 2 3 1 2 3 x

  where $x = a$, b or c → 4*3 combinations.

- $k = 2$

  $\rightarrow (n_r + 1) * (n_r + 2) * \frac{n_{new} * (n_{new} - 1)}{2}$ behaviors because once the first new product is ranked, you have $n_r + 1 + 1 = n_r + 2$ possible ranks for the second new product and you have $\frac{n_{new} * (n_{new} - 1)}{2} = \binom{n_{new}}{2}$ ways to choose 2 new products among the n_{new}. It doesn't matter which of the two products one considers to be the "first" because we obtain all possible orders, thus it would be the same if the "first" new product were to be the "second". This yields :

  

  where the arcs locate the possible ranks for the other new product. Thus we have 5*4 ranks combinations for each couple of new products and 3 possible couples: (a,b), (a,c) and (b,c), meaning that there are 60 possible coherent behaviors where two new products are ranked.

- thus for k

  $\rightarrow \prod_{i=1}^{k} (n_r + i) * \frac{\prod_{j=0}^{k-1} (n_{new} - j)}{k!} = \frac{(n_r + k)!}{n_r!} * \binom{n_{new}}{k}$

Finally, we obtain :

$$C = \sum_{k=0}^{n_{new}} \frac{(n_r + k)!}{n_r!} * \binom{n_{new}}{k} + 1$$

APPENDIX B INFORMATION THEORY: ENTROPY

We give an insight of why categorical data is more useful (which means that they convey more information) when the associated probability distribution is balanced. We will use the formalism developed by Boltzmann in a very concise way.

In Information theory, the entropy for a discrete random variable X with possible values ($X1, X2, ...X_n$) and the probability distribution associated ($p_1, p_2, ...p_n$) is defined as:

$$H(X) = E\left[-ln(P(X))\right] = -\sum_i p(x_i) ln(p_i)$$

In the very particular case where we consider only $n = 2$ variables, we have:

$$H(X) = -p ln(p) - (1-p) ln(1-p)$$

We can differentiate twice this function with regard to p:

$$\frac{\delta H(X)}{\delta p} = -ln(p) - \frac{p}{p} + ln(1-p) + \frac{1-p}{1-p} = ln(\frac{1-p}{p})$$

$$\frac{\delta^2 H(X)}{\delta p^2} = -\frac{1}{p^2}\frac{p}{1-p} = -\frac{p}{1-p}$$

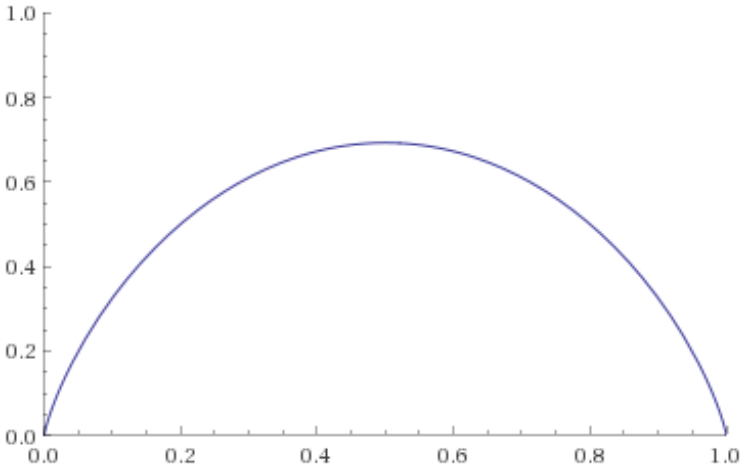

Figure B.1 Entropy for a discrete categorical variable with two outcomes, depending on the probability of the first outcome

The first order condition gives us the extremum: $ln(\frac{1-p}{p}) = 0 \implies p = \frac{1}{2}$, while the second order condition ($\frac{\delta^2 H(X)}{\delta p^2} < 0$) tells us that this extremum is a maximum. Therefore, the closer we are to the balanced distribution, the more information we have to exploit. We plotted in Figure B.1 the entropy H in function of the probability of the first outcome x (the second being $1-x$).

When considering $n = 3$ variables, the entropy becomes:

$$H(X) = -xln(x) - yln(y) - (1-x-y)ln(1-x-y)$$

We could show analytically that the maximum is also achieved when $x = y = z = \frac{1}{3}$. The plot giving the entropy H in function of the two first outcomes x and y (the third outcome being necessarily $1-x-y$) is in Figure B.2. We can notice that the maximum is achieved for $x = y = z = \frac{1}{3}$.

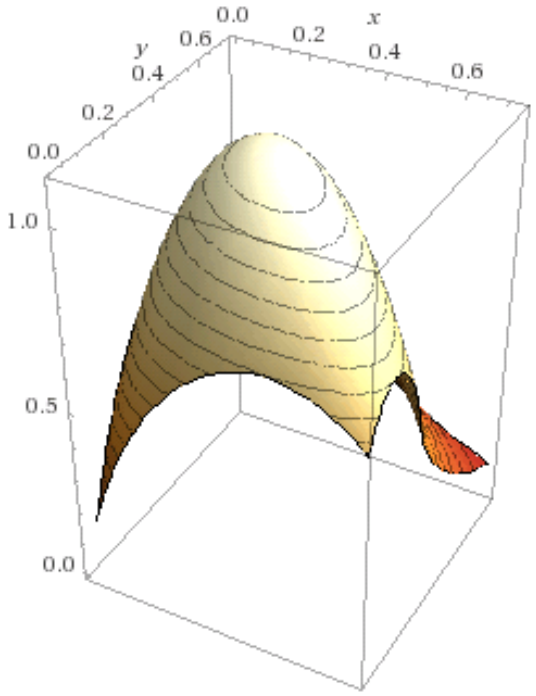

Figure B.2 Entropy for a discrete categorical variable with three outcomes, depending on the probability of the two first outcome x and y

This result can be generalized: categorical data with n possible outcomes is more useful when the outcomes probabilities are balanced. In our project, we noticed that the categories of the Shirts dataset are more balanced, as well as more numerous, than the Shoes dataset. Therefore, the features of the products of the Shirts set convey more information, and are more likely to be useful.